TESIS DOCTORAL

Un framework para el despliegue y evaluación de procesos software

Iván Ruiz Rube

Director:
Juan Manuel Dodero Beardo

UCA
Universidad de Cádiz

Universidad de Cádiz
Escuela Superior de Ingeniería
Doctorado en Ingeniería y Arquitectura

TESIS DOCTORAL

Un framework para el despliegue y evaluación de procesos software

AUTOR

Iván Ruiz Rube

DIRECTOR

Dr. Juan Manuel Dodero Beardo

Cádiz, 2013

Conformidad del director de tesis

D. Juan Manuel Dodero Beardo, profesor del Departamento de Ingeniería Informática de la Universidad de Cádiz, siendo director de la tesis titulada *Un framework para el despliegue y evaluación de procesos software*, realizada por el doctorando D. Iván Ruiz Rube dentro del programa de doctorado en Ingeniería y Arquitectura, para proceder a los trámites conducentes a la presentación y defensa de la tesis doctoral arriba indicada, informa que se autoriza la tramitación de la tesis.

El director de tesis

Juan Manuel Dodero Beardo

En Cádiz, España, a 14 de Noviembre de 2013

Agradecimientos

A Juanma, por darme la oportunidad de conocer este mundo de la universidad y de la investigación, y ofrecerme siempre los mejores consejos para que esta tesis pudiese llegar a buen puerto.

A Nacho, Juan y José Manuel, por haber implementado las ideas que se describen en este trabajo.

A mis compis, Miguel y Dani, y al resto de compañeros del grupo de investigación, por su apoyo.

A mis amigos, en especial a Moi, por recordarme siempre que hay vida más allá del trabajo.

A toda mi familia, especialmente a mi hermana por sus ánimos; a mi hermano, por haberme contagiado el gusanillo por los ordenadores; a mi madre, por su entrega infinita; y a mi padre, que seguro que estaría orgulloso de mí.

A Laura, por estar siempre a mi lado.

Gracias a todos.

Iván

Resumen

La Ingeniería de Procesos Software promueve la producción sistemática de software mediante el seguimiento de una serie de procesos bien definidos. Una gestión integral de dichos procesos implica el desarrollo de una serie de actividades como son el diseño de los modelos de procesos, la verificación, la validación, el despliegue y la posterior evaluación. El consorcio OMG publicó el estándar Software Process Engineering Metamodel (SPEM), un lenguaje destinado a facilitar y potenciar el entendimiento, la reutilización y la mejora de los procesos software. Después de realizar una revisión de la literatura con respecto a los usos del lenguaje, se pudieron extraer diversas conclusiones. La más importante es que el estándar ha tenido poca aceptación en la industria, en parte debido a la propia complejidad del lenguaje, a ciertas carencias existentes en aspectos como la gestión de la variabilidad de los procesos y su ejecutabilidad, y la falta de mecanismos para la automatización del despliegue sobre herramientas de soporte. Además, la evaluación de los procesos software es una actividad manual y su automatización requiere mejorar considerablemente la interoperabilidad entre las herramientas de apoyo a la producción y gestión del software.

Con los objetivos de minimizar los tiempos requeridos para adaptar las herramientas al comenzar cada nuevo proyecto y disminuir la complejidad técnica existente a la hora de construir mecanismos para automatizar la evaluación, se presenta Software Process Deployment & Evaluation Framework (SPDEF), un marco de trabajo para el despliegue y evaluación de procesos software. Este marco de trabajo se basa en la aplicación de las técnicas de la Ingeniería del Software dirigida por modelos y de la integración de información mediante datos abiertos enlazados. Utilizando las primeras, se consigue la adaptación semi-automática de las herramientas de soporte mediante la transformación sucesiva de modelos, partiendo desde el modelo de procesos. Con los datos abiertos enlazados, se consigue que las herramientas expongan de manera controlada la información que gestionan, para así facilitar la construcción de soluciones de integración destinadas a la evaluación de los procesos. El framework incluye, además de un método sistemático para el despliegue y evaluación, un conjunto de modelos y relaciones, así como una serie de herramientas de apoyo.

Para la evaluación del framework se han desarrollado dos casos de estudio consistentes en el despliegue de la metodología OpenUP sobre herramientas de soporte y en la evaluación de competencias en recursos de personal implicados en los procesos software. Además, se presenta un escenario detallado de integración para ilustrar cómo es posible automatizar las revisiones técnicas de calidad sobre los proyectos de desarrollo o mantenimiento de software.

Palabras clave: *Calidad del Software, Ingeniería de Procesos Software, Ingeniería del Software Dirigida por Modelos, Integración de Información, Datos Abiertos Enlazados.*

Abstract

Software Process Engineering promotes the systematic production of software by following a set of well-defined technical and management processes. A comprehensive management of these processes involves the accomplishment of a number of activities such as model design, verification, validation, deployment and evaluation.

The OMG consortium published the Software Process Engineering Metamodel (SPEM) intended to facilitate the understanding and sharing of software processes and so promoting its reuse and improvement. However, the current standard has not been widely recognized in the industry, in part due to the language complexity itself, some weaknesses in aspects such as the management of process variability and its executability, and the lack of mechanisms for the automated deployment into supporting tools. In addition, the automated evaluation of software processes is a complex activity due to the absence of interoperability between tools used to manage, develop or maintain software.

With the aim of minimizing the required time to adapt the tools in the beginning of each new project and reducing the high complexity for the construction of mechanisms for the automated evaluation, the Software Process Deployment & Evaluation Framework (SPDEF) has been elaborated. This framework is based on the application of well-known techniques in Software Engineering, such as Model Driven Engineering and Information Integration through Linked Open Data. Using the former, we can automatically customize the supporting tools by applying a number of model transformations, starting from the software process model. With the second approach, the tools will expose their data in a controlled way, thereby facilitating the development of integration solutions intended for process evaluation. In addition to the method for the deployment and evaluation, this framework also includes a set of models, relationships between models and software tools.

Two case studies are presented as evaluation of the proposed framework. First, the deployment of the OpenUP methodology into several supporting tools and second, the skill assessment of human resources involved in software processes. Finally, a detailed integration scenario for assessing quality in software development or maintenance projects is also included.

Keywords: *Software Quality, Software Process Engineering, Model-driven Engineering, Information Integration, Linked Open Data.*

Índice general

Índice de tablas

Índice de figuras

Índice de listados de código

Parte I

Prolegómeno

Capítulo 1

Introducción

En este capítulo, se presenta una introducción al trabajo de investigación llevado a cabo en el ámbito de la mejora de los procesos software. Se describe la motivación del mismo, sus objetivos y el método de investigación empleado. Finalmente, se detalla la estructura del resto del documento.

1.1. Motivación

La Ingeniería del Software consiste en la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software [34]. Actualmente, la principal guía de conocimiento en Ingeniería del Software se encuentra en el proyecto *Software Engineering Body of Knowledge* (SWEBOK) [24]. Este proyecto desarrolló un documento que define las diez áreas de conocimiento fundamentales en Ingeniería del Software y un conjunto de disciplinas interrelacionadas. De las áreas de conocimiento de SWEBOK, la Calidad de Software es ubicua al resto de áreas y consecuentemente a toda la Ingeniería del Software. Existen diferentes definiciones del concepto de calidad. Una de las más significativas es la propuesta por la *International Organization for Standardization* (ISO):

> La calidad es el grado en el que un conjunto de características inherentes cumple con los requisitos establecidos [140].

La gestión de la calidad tiene por objetivo planificar, asegurar, controlar y mejorar la calidad del software que se desarrolla o se mantiene. Sin embargo, el

término calidad de software no sólo hace referencia a la calidad del producto, sino también a la calidad de los propios procesos de desarrollo o mantenimiento del software, a la calidad de los recursos empleados (herramientas, técnicas, métodos o guías) y a las habilidades de las personas implicadas. Es una premisa que la calidad del producto software viene directamente influenciada por la calidad en la ejecución de las actividades definidas en los procesos organizativos [87]. Por este motivo, los esfuerzos relativos a la calidad deben focalizarse no sólo a nivel de producto, sino también a nivel de proceso.

La Ingeniería de Procesos Software es el área de la Ingeniería del Software que promueve la producción sistemática de software mediante el seguimiento de procesos técnicos y de gestión bien definidos, con el objetivo de maximizar la calidad [53]. Así pues, las organizaciones necesitan disponer de métodos, técnicas y herramientas que les permitan aplicar una estrategia integral de mejora continua de la calidad en sus procesos de desarrollo o mantenimiento de software. Una gestión integral de los procesos software requiere de instrumentos para el diseño, verificación, validación, despliegue y evaluación de los procesos. Sin embargo, las herramientas de soporte no suelen ofrecer mecanismos para incorporar las definiciones explicitas de los procesos, lo cual provoca una falta importante de consistencia entre los modelos de procesos y su despliegue real sobre las herramientas.

Por otra parte, en los últimos años han ido apareciendo multitud de métodos, buenas prácticas, técnicas y herramientas, todas ellas orientadas a mejorar las actividades de gestión y producción del software. Esto se debe, en parte, al crecimiento exponencial que ha tenido Internet y a la gran aceptación que tienen las tecnologías de la información en todos los ámbitos de la sociedad. Así, es habitual ver cómo en diferentes foros se promueve el uso de determinadas técnicas, herramientas o métodos sobre otros, aludiendo a simples convicciones, experiencias personales o sobre la base de estrategias comerciales o de marketing. Podemos citar algunos ejemplos en este sentido: ¿realmente la programación orientada a objetos facilita el mantenimiento del software, en comparación con la programación estructurada?, ¿el desarrollo dirigido por pruebas permite construir aplicaciones de mayor calidad y en menos tiempo? o ¿cuál de los numerosos *frameworks Java* para la construcción de aplicaciones web permite una mayor reducción de los tiempos de desarrollo? En este sentido, Meyer [126] recapituló muchas de estas cuestiones fundamentales que requieren de respuestas empíricas, creíbles y útiles para profesionales, académicos e investigadores en Ingeniería del Software.

En muchas ocasiones es complicado afirmar con certeza si una técnica, herramienta o método de Ingeniería ayuda a mejorar los tiempos de desarrollo o

la calidad del software resultante. De este modo surge la Ingeniería del Software Empírica [21], con el objetivo de llevar a cabo evaluaciones formales sobre las técnicas, lenguajes, herramientas u otros elementos. Esta iniciativa promueve acercar la Ingeniería del Software a la Ciencia, de forma que mediante el desarrollo de estudios y experimentos empíricos, podamos analizar y discernir sobre aspectos diversos en la gestión y construcción del software. La experimentación en Ingeniería del Software es una disciplina relativamente reciente, cuyo objetivo es buscar respuestas cuantitativas a preguntas concretas.

Para realizar las experimentaciones formales se necesitan evidencias y datos sobre los procedimientos, herramientas y recursos utilizados para desempeñar las actividades necesarias en los proyectos. A partir de los datos y de las evidencias generadas en las diferentes herramientas de soporte al proceso software, se podrían analizar indicadores que permitan exponer las bondades y debilidades de los elementos intervinientes en los procesos de desarrollo, operación o mantenimiento de software. Sin embargo, la implantación de procedimientos de evaluación automáticos sobre dichos elementos es una tarea de elevada complejidad.

Esta tesis propone un marco de trabajo para el despliegue y evaluación de los procesos software. Para ello, se aplican las técnicas de la Ingeniería del Software dirigida por modelos o *Model-Driven Engineering* (MDE), para automatizar el despliegue de las definiciones de procesos sobre herramientas de soporte, y las técnicas de datos abiertos enlazados o *Linked Open Data* (LOD), para desarrollar soluciones de integración de información dirigidas a la evaluación de los procesos software, utilizando los datos gestionados en las distintas herramientas. Estas técnicas ya han demostrado aportar importantes beneficios para la comunidad de Ingeniería del Software. La primera de ellas, MDE, se centra en el diseño y transformación de modelos para la mejora de la productividad durante el desarrollo del software, mientras que LOD, permite simplificar el diseño de los procesos de integración de datos necesarios para publicar y consumir datos de sistemas en la Web.

1.2. Objetivos

La presente tesis doctoral, focalizada en la mejora de la calidad en la gestión de procesos software, pretende satisfacer los siguientes objetivos concretos:

- *OBJ1. Recopilar el estado del arte actual en la gestión de procesos software.* Para ello, se realizará una metódica revisión de la literatura con

respecto a la gestión de procesos software, desde la perspectiva del ciclo de vida de los procesos de negocio.

- *OBJ2. Posibilitar la automatización de los procesos de despliegue de procesos software sobre herramientas de soporte.* Para satisfacer este objetivo se elaborarán un conjunto de pautas, modelos y transformaciones para la ejecución controlada de proyectos utilizando herramientas adaptadas a las metodologías de software.

- *OBJ3. Mejorar los procedimientos necesarios para la evaluación de la calidad en los procesos software.* Para alcanzar este objetivo se plantearán una serie de recomendaciones para la publicación y consumo de los datos procedentes de las herramientas de soporte, con el fin de construir soluciones de integración de datos que faciliten la implementación de procedimientos automatizados de evaluación de procesos.

1.3. Método

La presente memoria de tesis doctoral, último paso para alcanzar el grado de *Doctor en Ingeniería y Arquitectura* por la *Universidad de Cádiz*[1], está focalizada en el ámbito de la *Mejora de Procesos Software*, dentro del área de conocimiento *Lenguajes y Sistemas Informáticos*.

Para la elaboración de esta tesis, dedicada al despliegue y evaluación de procesos software, se toma como base el modelo descrito por Oates [134] con el fin de obtener un trabajo original de investigación, accesible y comprensible sobre el dominio del problema en cuestión. A partir de este proceso general, se diseñó un plan de trabajo específico (véase el cronograma de la figura 1.1) para el desarrollo de esta tesis a lo largo de 4 años (12 cuatrimestres). Este plan se compone de seis actividades, de las cuales, la actividad *A1* se encuadra en el cumplimiento del objetivo *OBJ1*, mientras que el resto de actividades contribuye de forma conjunta tanto para el objetivo *OBJ2*, como para el objetivo *OBJ3*. A continuación, se describen cada una de las actividades del plan de trabajo.

A1. Revisión de la literatura sobre la gestión de procesos software: Esta actividad tiene por objetivo conocer el nivel de aceptación del lenguaje *Software and Systems Process Engineering Metamodel Specification* (SPEM) como

[1] http://www.uca.es

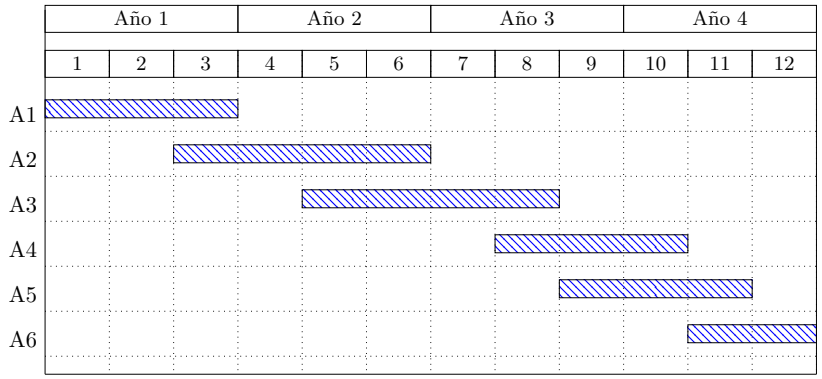

Figura 1.1: Plan de trabajo

medio para la representación de procesos software, así como exponer los beneficios reales que proporciona. Para el desarrollo de esta revisión de la literatura, se utilizaron el método propuesto por Kitchenham [94] y las directrices de Petersen [143] para planificar, ejecutar y presentar los resultados de un estudio de alcance o *Systematic Mapping Study* (SMS). Estas pautas están teniendo cada vez mayor aceptación en el desarrollo de revisiones de la literatura en Ingeniería del Software. El proceso comienza con la justificación de la necesidad de realizar tal estudio, así como la formulación de las cuestiones de investigación a las que se quieren ofrecer respuesta. Luego, se lleva a cabo un riguroso proceso de búsqueda de los trabajos relacionados, seleccionando aquellos de mayor interés y descartando aquellos que cumplan alguno de los criterios de exclusión planteados. Finalmente, se genera un esquema de clasificación a partir de los datos extraídos de los estudios primarios seleccionados, para posteriormente analizar los resultados.

A2. Revisión del estado actual de los paradigmas MDE y LOD: La contribución principal de la tesis se basa en la aplicación de unos determinados paradigmas metodológicos y técnicos al problema del despliegue y evaluación de procesos. Por ello, se requiere el estudio de los paradigmas empleados, a saber, la Ingeniería del Software dirigida por modelos o MDE y la integración de información mediante datos abiertos enlazados o LOD.

A3. Diseño de un método para el despliegue y evaluación de procesos software: En esta actividad se elabora un método con el cual abordar algunos de los problemas detectados con la actividad *A1*, esto es, la falta de alineamiento entre los procesos y las herramientas, y la falta de automatización en la evaluación de los procesos software. Este marco de trabajo o *framework*[2] se compone de un método para el despliegue y evaluación de los procesos sobre herramientas de soporte a la gestión o producción del software.

A4. Implementación de los componentes software necesarios para dar soporte al marco propuesto: Además del método y de los modelos necesarios para el despliegue y evaluación de los procesos software, se requiere la implementación de una serie de herramientas que permitan asegurar la factibilidad técnica de la propuesta. Con esta actividad se completan los elementos que forman parte del marco de trabajo.

A5. Evaluación del framework propuesto: Con el objetivo de demostrar la utilidad, calidad y eficacia del *framework* propuesto, es necesario aplicar diversos mecanismos para su evaluación, como los que se presentan en [78]. Con respecto al despliegue de procesos, se realiza un caso de estudio consistente en desplegar los productos de trabajo identificados en una determinada metodología de software, sobre una serie de herramientas de soporte. De cara a la evaluación de procesos, se presenta un caso de estudio relativo al análisis de indicadores, extraídos de las herramientas de soporte, sobre el desempeño de los miembros participantes en los equipos de proyecto. Asimismo, se describe un escenario descriptivo de integración de datos para la realización de revisiones técnicas de calidad sobre determinados aspectos de Ingeniería del Software en proyectos de desarrollo.

A6. Formulación de conclusiones: Esta última actividad tiene por objetivo analizar el cumplimiento de los objetivos planteados con la tesis y detectar futuras líneas de investigación.

[2] Un *framework* se define como la estructura conceptual básica, o conjunto de ideas o hechos, que proporciona soporte para abordar un tipo de problemática en particular, que sirve como referencia para resolver nuevos problemas de índole similar.

1.4. Estructura del documento

Este documento representa el resultado principal del trabajo de investigación y se estructura de la siguiente forma:

En la primera parte de la tesis, formada por este capítulo a modo de prolegómeno, se ofrece una introducción al trabajo y se resumen los principales objetivos de la investigación.

En la segunda parte se presenta el estado de la cuestión, donde se describen los antecedentes necesarios para una correcta comprensión del contexto de la investigación. En el capítulo 2 se presentan los lenguajes y herramientas para el modelado de procesos, mientras que en el capítulo 3 se presenta la revisión de la literatura llevada a cabo acerca de la gestión de procesos software.

La tercera parte de la tesis se centra en el marco de trabajo para el despliegue y evaluación de procesos software. En el capítulo 4 se detalla el problema que origina su desarrollo; el capítulo 5 resume los fundamentos metodológicos y técnicos que lo sustentan; el capítulo 6 presenta los detalles del *framework* y en el capítulo 7 se describe cómo se ha llevado a cabo su evaluación.

En el epílogo (capítulo 8) se recogen las conclusiones, las líneas de trabajo futuras y las contribuciones desarrolladas a partir de esta investigación. Como anexos, se incluyen el conjunto de tablas de clasificación de las publicaciones analizadas para el estudio descrito en el capítulo 3 y una serie de listados de código relativos a la implementación del *framework*. Finalmente, se incluyen el glosario de siglas utilizadas a lo largo del documento y las referencias empleadas durante el desarrollo del trabajo.

Parte II

Estado de la cuestión

Capítulo 2

Lenguajes y herramientas de modelado de procesos

La Ingeniería de Procesos Software, pilar fundamental de la presente tesis doctoral, promueve la producción sistemática de software mediante el seguimiento de procesos técnicos y de gestión bien definidos. En este capítulo se describen los lenguajes más importantes para el modelado de procesos software, haciendo especial énfasis en el estándar SPEM y las herramientas de soporte al modelado.

2.1. Lenguajes de modelado de procesos software

Hoy día, las organizaciones dedicadas parcial o totalmente al desarrollo de software suelen implantar algún tipo de modelo de proceso o metodología para mejorar la calidad de los productos o servicios ofrecidos. La calidad del software depende en gran medida de la calidad del proceso de desarrollo empleado. Por ello, metodologías tradicionales como *Rational Unified Process* (RUP) [101] y ágiles como *Scrum* [165] están adquiriendo un papel cada vez más importante en la industria del software. Con independencia de la metodología o modelo implementado, es común la estrategia para la mejora continua de la calidad, basada en el Círculo de Deming o *Plan, Do, Check, Act* (PDCA) [45]. Según esta estrategia, todo proceso debe planificarse, implantarse y evaluarse, para luego poder actuar sobre él.

Los procesos software son conjuntos coherentes de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software [63]. La definición explicita de los procesos juega un papel fundamental en las principales iniciativas para la mejora de procesos software. Por ejemplo, en *Capability Maturity Model Integration* (CMMI) [169] la definición de procesos se incluye dentro del área *Organizational Process Definition*, mientras que en ISO/IEC 12207 [175] se incluye en el grupo de procesos *Process Improvement*.

Los lenguajes de modelado permiten construir descripciones de procesos de forma homogénea, utilizando habitualmente una notación gráfica. Estos lenguajes ayudan a mejorar la correcta comprensión de los procesos por parte de todos los implicados en el mismo. En [141] se describen los principales lenguajes para el modelado de procesos, entre los que se incluyen *Business Process Modeling Notation* (BPMN), *XML Process Definition Language* (XPDL), *JBPM Process Definition Language* (JPDL), *Architecture of Integrated Information Systems* (ARIS), *Integration DEFinition* (IDEF) y los diagramas de actividad de *Unified Modeling Language* (UML). Además de los anteriores, se han desarrollado otros lenguajes específicos para el modelado de procesos software, los cuales comparten ciertas características comunes, entre otras:

- Actividades, como el diseño de la arquitectura del software.
- Recursos, como una herramienta de gestión documental.
- Productos de trabajo, como un catálogo de requisitos.
- Actores, como un responsable de pruebas o *tester*.
- Reglas, como que los requisitos deben ser aceptados por el responsable del cliente antes de seguir con las siguientes etapas de desarrollo.

A continuación, se presentan algunos de los lenguajes más extendidos para el modelado de procesos software.

2.1.1. OPF

Open Process Framework (OPF) es una propuesta libre (dominio público) basada en la industria para la producción sistemática de metodologías de desarrollo [60]. OPF se compone de un metamodelo de procesos, un conjunto de métodos reutilizables y un conjunto de directrices para la creación de nuevas metodologías o adaptar metodologías ya existentes. En la figura 2.1, podemos comprobar las clases principales del metamodelo de OPF.

2.1.2. SEMDM

En el estándar ISO/IEC 24744 se define *Software Engineering Metamodel for Development Methodologies* (SEMDM), una alternativa para el modelado de procesos de desarrollo software [83]. Este estándar internacional hace uso de un nuevo enfoque para definir metodologías, basándose en el concepto de *powertypes* y *clabjects*. Para ello, se basa en una arquitectura de metamodelado de tres capas (*Endeavour Domain*, *Methodology Domain* y *Metamodel Domain*), en lugar de la más extendida arquitectura de cuatro capas propuesta por el consorcio *Object Management Group* (OMG). En la figura 2.2 se muestran las clases principales del metamodelo SEMDM.

2.1.3. MSF

Microsoft Solution Framework (MSF) es el marco de trabajo que propone *Microsoft* para definir procesos software. El metamodelo de MSF se compone de una serie de principios fundamentales, un modelo de equipo, fases e iteraciones. De esta forma, MSF soporta múltiples enfoques de procesos, los cuales pueden adaptarse a cualquier proyecto con independencia de su complejidad o tamaño. En la figura 2.3, podemos observar una representación abstracta del metamodelo MSF, extraída de [185].

2.2. El lenguaje SPEM

El consorcio OMG, conocido internacionalmente por la especificación UML, publicó en el año 2002 la especificación SPEM, un lenguaje diseñado para modelar procesos de Ingeniería del Software. Sin duda, se trata del lenguaje de definición de procesos software más utilizado.

El lenguaje, actualmente en su versión 2.0, se ofrece en dos versiones: como metamodelo *Meta-Object Facility* (MOF) y como perfil UML. SPEM proporciona una serie de elementos para representar de forma estandarizada los métodos, ciclos de vida, roles, actividades, tareas y productos de trabajo utilizados en Ingeniería del Software. Así, a modo de ilustración, podemos afirmar que SPEM es a los procesos software lo mismo que UML es a los sistemas software. En la figura 2.4 se representan los elementos principales del lenguaje (con su icono representativo de la versión perfil UML), agrupados según correspondan a contenido de métodos (roles, tareas, productos de trabajo, etc.) o a la formación de procesos (roles en uso, tareas en uso, actividades, etc.).

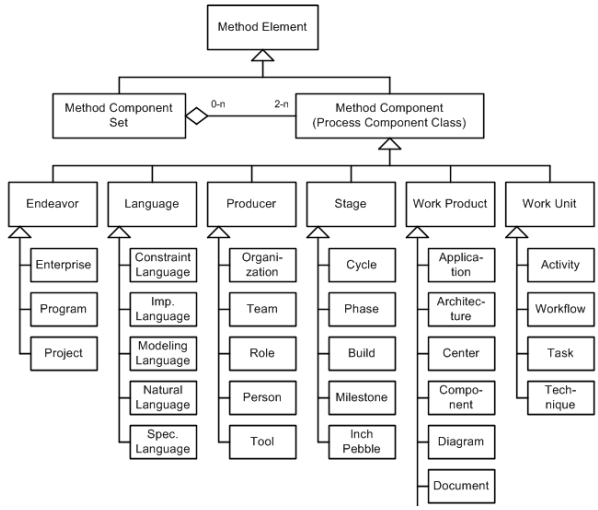

Figura 2.1: Elementos principales del metamodelo OPF

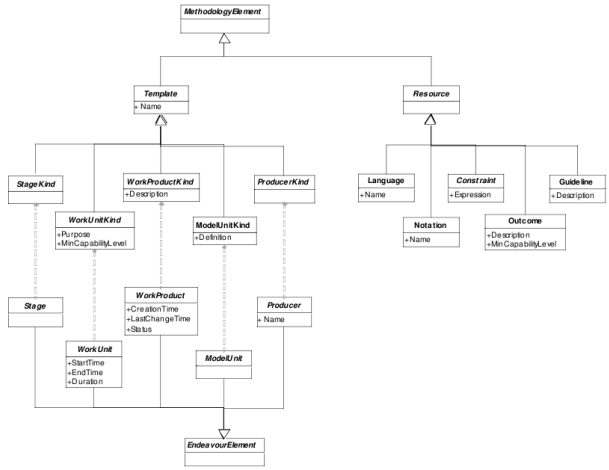

Figura 2.2: Elementos principales del metamodelo SEMDM

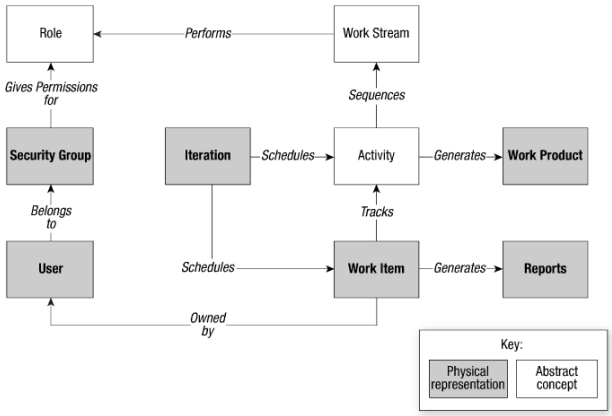

Figura 2.3: Elementos principales del metamodelo MSF

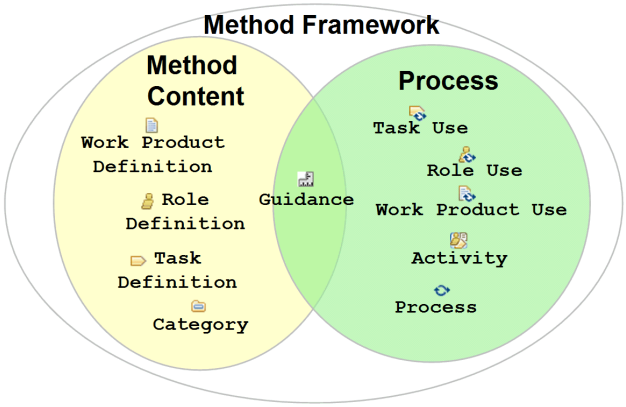

Figura 2.4: Elementos principales del metamodelo SPEM

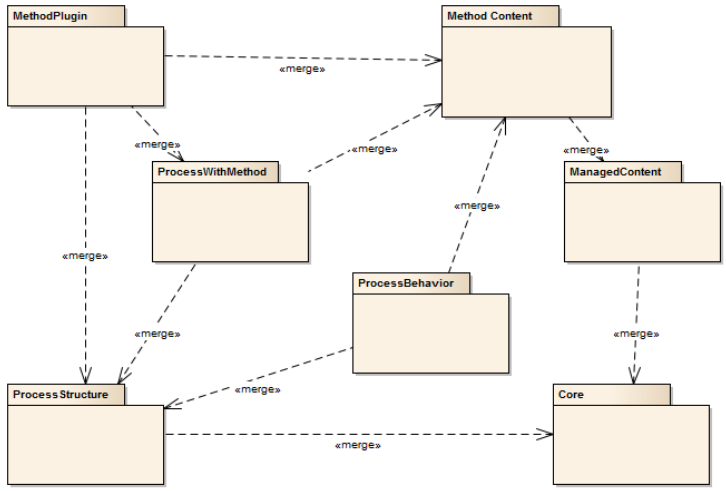

Figura 2.5: Paquetes del metamodelo de SPEM

En la especificación de SPEM [136] se describen los beneficios potenciales que se pueden obtener al modelar procesos software con este lenguaje. Son los siguientes:

- Proporcionar una representación estándar para los procesos y contenidos de métodos.
- Dar soporte al desarrollo sistemático y a la gestión de procesos de desarrollo.
- Permitir la adaptación de los procesos a las necesidades específicas de los proyectos, mediante el uso de extensiones, omisiones y puntos de variabilidad. Esta característica se denomina *process tailoring*.
- Dar soporte al despliegue para la ejecución automática de procesos, lo que se conoce como *process enactment*.

El metamodelo de la versión 2 de SPEM está compuesto por siete paquetes, como podemos apreciar en la figura 2.5. El metamodelo define un lenguaje muy

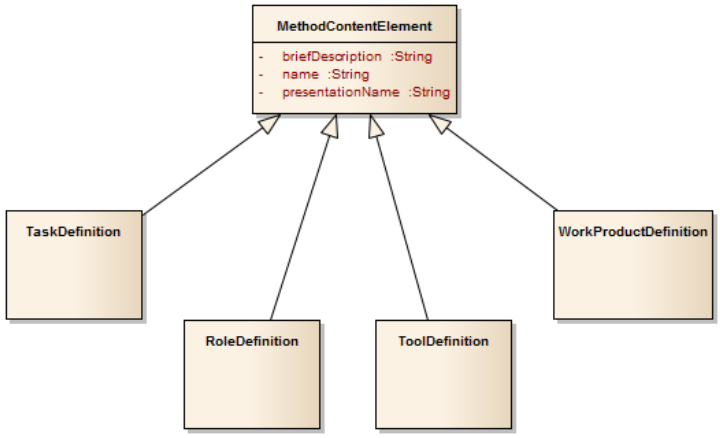

Figura 2.6: Contenidos de método en SPEM

amplio, con gran cantidad de metaclases y metaasociaciones. Por ello, a continuación se detallan sólo los paquetes principales y los elementos con interés para las contribuciones que se presentan en esta tesis doctoral.

2.2.1. Method Content

En este paquete se incluyen los constructores necesarios para dotar de contenido a las metodologías de software, definiendo enunciados del tipo: un determinado miembro del equipo de proyecto (*Role Definition*) realiza una determinada tarea (*Task Definition*), utilizando unas entradas (*Work Product Definition*) para obtener unas salidas (*Work Product Definition*).

En la figura 2.6 podemos comprobar cómo la metaclase abstracta *MethodContentElement* se especializa en *TaskDefinition*, *RoleDefinition*, *ToolDefinition* y *WorkProductDefinition*, los elementos nucleares del lenguaje SPEM. En ésta y posteriores figuras se omiten muchos de los atributos de las metaclases, a excepción de aquellos que son más significativos. Asimismo, y por motivos de simplicidad, se omiten también varias de las metaclases y asociaciones que aparecen en la especificación oficial de la OMG.

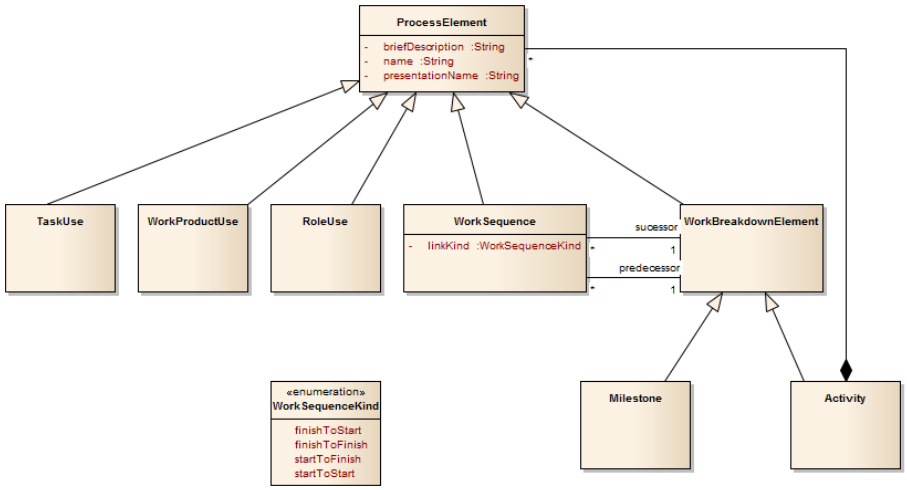

Figura 2.7: Estructura de procesos en SPEM

2.2.2. Process Structure

El paquete *Process Structure* ofrece los constructores necesarios para definir la estructura de desglose de trabajo de los procesos, mediante la secuenciación de actividades. En otras palabras, este paquete permite definir modelos de procesos.

Un extracto del contenido de este paquete se presenta en la figura 2.7. En este paquete, la metaclase *ProcessElement* (generalización de cualquier elemento que forme parte de un proceso SPEM) se especializa en otras como *TaskUse*, *WorkProductUse* y *RoleUse*, las cuales representan la utilización efectiva de una tarea, un producto de trabajo o un determinado rol, en el contexto de un determinado proceso. Al mismo tiempo, *ProcessElement* se especializa en elementos *WorkBreakdownElement* (generalización de los elementos que forman parte de un desglose de trabajo) y *WorkSequence* (representación de una relación de dependencia entre dos *WorkBreakdownElement*).

Asimismo, un elemento de tipo *WorkBreakdownElement* puede especializarse en un *Milestone* para representar un hito importante dentro de un proyecto o en un elemento *Activity* como la unidad básica de trabajo. Nótese como este último concepto puede incluir a su vez a otros *ProcessElement*.

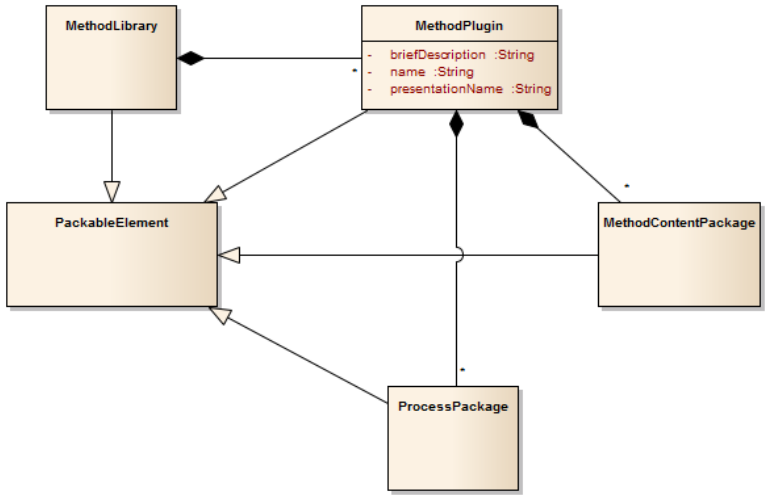

Figura 2.8: Plugin de métodos en SPEM

2.2.3. Method Plugin

El paquete *MethodPlugin* nos ofrece los elementos necesarios para gestionar repositorios de contenidos de métodos y de procesos, así como dar soporte a las necesidades de reutilización y adaptación de procesos para diferentes entornos. El elemento contenedor principal es el *MethodLibrary*, que se compone de elementos *MethodPlugin*. Cada uno de los *plugins* de método se compone de elementos *ContentMethodPackage* y *ProcessPackage*, los cuales se comportarán como contendedores de contenidos de método (definiciones de roles, tareas, etc.) y de procesos (secuenciación de actividades). Podemos observarlo en la figura 2.8.

La especificación del lenguaje SPEM incluye un *plugin* de métodos predeterminado, el cual proporciona instancias muy comunes en metodologías de software. Este *plugin*, *SPEM 2.0 Base Plug-in*, proporciona un conjunto de clases que permiten especializar las actividades, los productos de trabajo y otros elementos de contenido SPEM, tal y como se describe a continuación.

El elemento *Activity* (véase la figura 2.9) se especializa en *Phase*, para declarar un periodo significativo de tiempo; *Iteration*, para representar ciclos repeti-

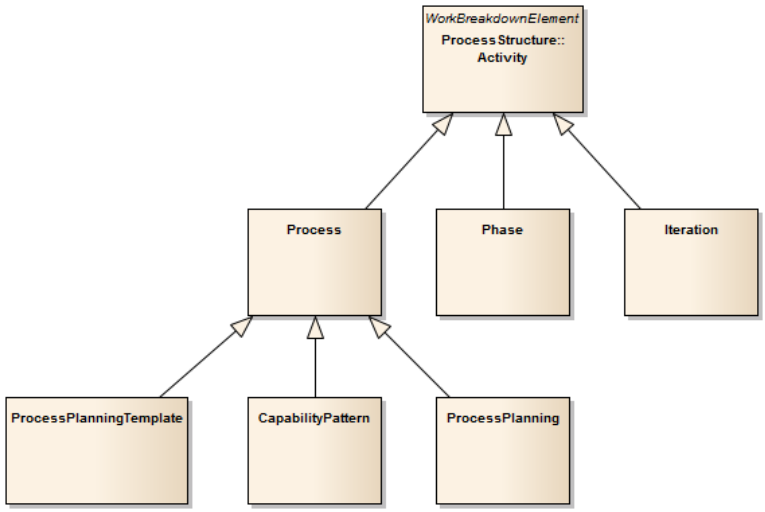

Figura 2.9: Tipos de actividades en SPEM

tivos; o *Process*, para definir la estructura de procesos de desarrollo de software. Los elementos *Process* a su vez pueden clasificarse en *DeliveryProcess*, para definir procesos completos; *ProcessPattern*, para definir partes de procesos reutilizables; o *ProcessPlanningTemplate*, definido como un proceso especial apto para ser instanciado en herramientas de planificación de proyectos.

Se consideran tres tipos de productos de trabajo en SPEM: *Outcome*, que proporciona una descripción para productos de trabajo no tangibles, por ejemplo para definir estados o resultados de un proyecto; *Artifact*, para identificar productos tangibles, como por ejemplo un modelo UML; y *Deliverable*, para representar productos de trabajo con valor para terceros, como por ejemplo un documento de requisitos. Lo podemos observar en la figura 2.10.

En la figura 2.11 se incluyen algunos descriptores para categorizar elementos de contenido SPEM: *RoleSet*, que permite categorizar roles (analistas, diseñadores, etc.); *ToolCategory*, para clasificar herramientas (entornos de desarrollo, herramientas de gestión, etc.); *Domain*, para categorizar productos de trabajo según su utilidad (arquitectura, operaciones, etc.); y *Discipline*, para categorizar tareas según su ámbito (análisis, diseño, etc.).

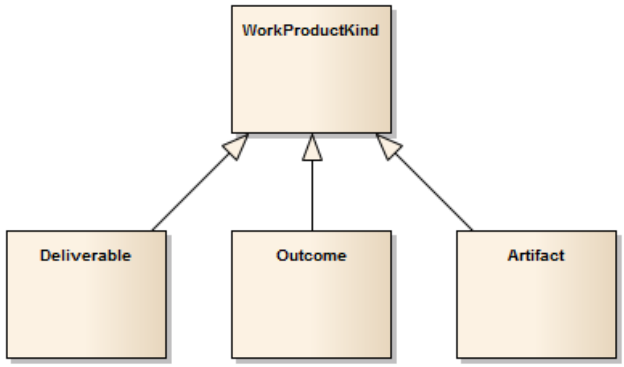

Figura 2.10: Tipos de productos de trabajo en SPEM

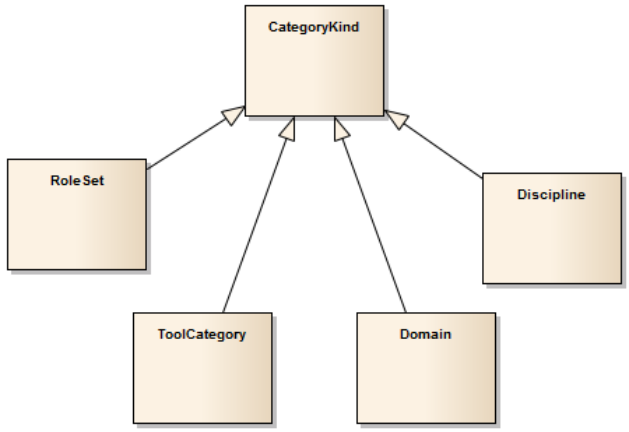

Figura 2.11: Tipos de categorías en SPEM

2.2.4. Otros paquetes

Además de los paquetes descritos anteriormente, el metamodelo SPEM incluye otros, no menos importantes, que brevemente se describen a continuación.

- *Core:* Se trata del paquete central del metamodelo, el cual contiene el conjunto de constructores y clases abstractas necesarias para el resto de paquetes.
- *Managed Content:* Este paquete permite incorporar documentos y descripciones en lenguaje natural en aquellas situaciones donde no es posible expresar ciertos conceptos mediante modelos estructurados. Las guías pueden ser plantillas, listas de comprobación, documentos de buenas prácticas, etc.
- *Process Behavior:* Permite extender los modelos de procesos con modelos de comportamiento externos, como por ejemplo, los diagramas de actividad de UML.
- *Process with Methods:* Los elementos de este paquete permiten integrar los modelos de procesos con los contenidos de método definidos mediante los elementos de *Process Structure* y *Method Content*.

2.3. Herramientas de soporte al lenguaje SPEM

Para una correcta definición de los modelos de procesos es preciso disponer de herramientas que ofrezcan soporte a los lenguajes de modelado. En esta sección se recogen las herramientas principales de soporte al lenguaje SPEM.

2.3.1. EPF Composer

Eclipse Process Framework (EPF) *Composer*[1] es la herramienta de la comunidad *Eclipse* para la creación, edición y mantenimiento de fragmentos de métodos, procesos o metodologías de Ingeniería del Software. La herramienta está basada en el estándar SPEM 2, aunque en realidad, utiliza una versión adaptada del metamodelo, denominada *Unified Method Architecture* (UMA). En la figura 2.12 podemos visualizar una captura de pantalla de la herramienta.

[1] http://www.eclipse.org/epf

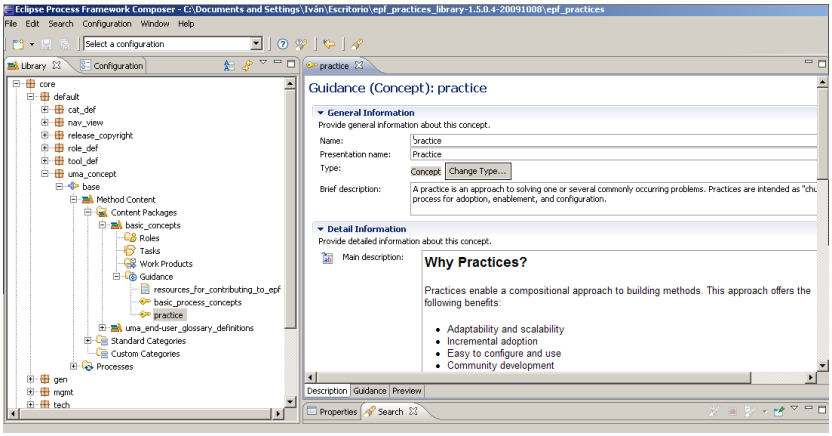

Figura 2.12: Entorno de modelado SPEM en EPF

2.3.2. IRIS Process Author

La compañía *Osellus* dispone de la herramienta comercial *IRIS Process Author*[2], dedicada a la autoría de procesos software compatibles con SPEM. La herramienta ofrece la posibilidad de utilizar tecnologías wiki con el objetivo de revisar y refinar los activos de procesos. La figura 2.13 muestra el entorno de modelado de este software.

2.3.3. Enterprise Architect

Enterprise Architect[3], de *Sparx Systems* es una de las herramientas más completas para el diseño de sistemas software. La herramienta permite trabajar con diferentes lenguajes de modelado, entre ellos UML y SPEM. Además, este software permite gestionar la trazabilidad desde los requisitos al despliegue de sistemas, aplicar ingeniería directa e inversa de código y realizar ciertas actividades de gestión de proyectos. En la figura 2.14 se muestra una captura de pantalla de la herramienta.

[2] http://www.osellus.com/IRIS-PA
[3] http://www.sparxsystems.com

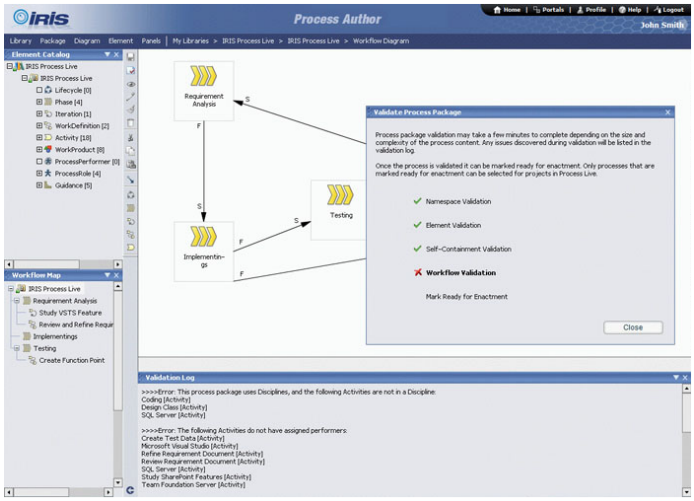

Figura 2.13: Entorno de modelado SPEM en IRIS Process Author

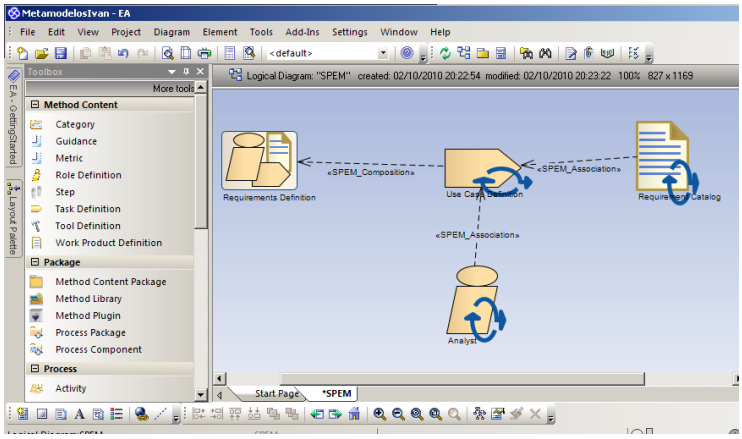

Figura 2.14: Entorno de modelado SPEM en Enterprise Architect

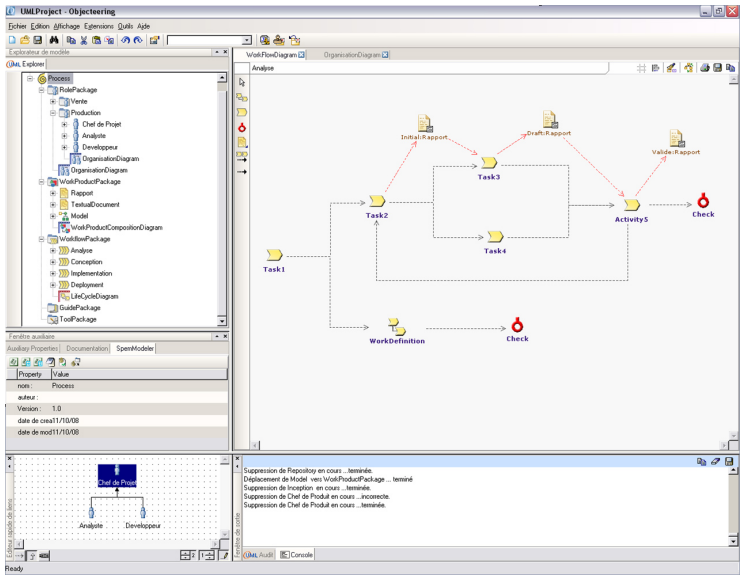

Figura 2.15: Entorno de modelado SPEM en Objecteering

2.3.4. Objecteering

Objecteering[4] es una herramienta de la compañía *Objecteering Software* basada en MDE, que permite realizar modelos estándares como UML y BPMN, gestionar catálogos de requisitos, chequear la consistencia de modelos y otras capacidades. Este entorno dispone de una extensión que permite modelar procesos software con SPEM. En la figura 2.15 se muestra una captura de pantalla de la herramienta.

[4] http://www.objecteering.com

Capítulo 3

Gestión de procesos software

En el capítulo anterior describimos, entre otros aspectos, el estándar de modelado SPEM para la definición de procesos software. Los beneficios potenciales del lenguaje parecen bastante prometedores: reutilización de métodos y procesos entre diversas organizaciones y sentar las bases para la automatización de los procesos, entre otros [136]. Esto último, haciendo uso de los principios, técnicas y herramientas que ofrece MDE. Sin embargo, ¿cuál ha sido el grado de aceptación del lenguaje?, ¿se han cumplido las expectativas puestas en él?, ¿es suficiente la especificación actual de SPEM o necesitaría ser extendida?, ¿se han desarrollado transformaciones desde modelos SPEM a otros modelos?, ¿cuáles han sido los beneficios reales a la hora de utilizar SPEM? Posteriormente, se detallarán de una manera más formal estas cuestiones.

Con el objetivo de dar respuesta a estas preguntas generales, se ha llevado a cabo un estudio de alcance para medir el nivel de aceptación del lenguaje SPEM como medio para la representación de procesos software y al mismo tiempo, exponer los usos y los beneficios reales que proporciona [160]. De esta forma, se pretende encontrar clústeres de evidencias y áreas deficitarias donde focalizar futuros esfuerzos de investigación. Otra razón fundamental que motivó el desarrollo de este estudio fue la ausencia, hasta la fecha, de revisiones de la literatura en este ámbito. El único estudio encontrado fue uno relativo a la construcción y adaptación de métodos para situaciones específicas en proyectos de desarrollo de software [76], la denominada *Situational Method Engineering*

(SME). Recientemente, otros autores han realizado una revisión sistemática de la literatura para recopilar los lenguajes de modelado de procesos software [65], donde han identificado algunos de los problemas que se describen en esta tesis, corroborando así los hallazgos encontrados.

En este capítulo, se describe el estado actual de la investigación en torno a la gestión de procesos software, desde la perspectiva del ciclo de vida de los procesos de negocio. Analizando el soporte que ofrecen los modelos de procesos SPEM para la gestión de los procesos software, se pretende satisfacer el primero de los objetivos de la tesis. A continuación, se describe el procedimiento empleado en el estudio, los resultados obtenidos y una discusión de los principales hallazgos encontrados.

3.1. Procedimiento de estudio

El procedimiento seguido para la realización del estudio de alcance está modelado con el propio lenguaje SPEM, utilizando para ello la herramienta EPF. Esta herramienta nos permite además, generar una documentación en formato HTML, de tal modo que el procedimiento pueda ser fácilmente reutilizado en futuros estudios de la literatura.

Es fundamental seguir un procedimiento sistemático a la hora de llevar a cabo una revisión de las evidencias empíricas existentes sobre algún área de interés. En las siguientes subsecciones se describen las cuestiones de investigación, la estrategia de búsqueda y los criterios de selección de estudios. Luego se explican los aspectos relacionados con el diseño del esquema de clasificación y la extracción de datos. Finalmente, se incluye la estrategia para la presentación y análisis de los datos obtenidos.

3.1.1. Cuestiones de investigación

Los procesos software son un tipo particular de procesos de negocio, que se definen como el conjunto de actividades que se desarrollan en coordinación y bajo un entorno organizacional y tecnológico, para alcanzar un determinado objetivo de negocio [191]. De esta forma, los procesos software se basan en una colección de actividades relacionadas entre sí, cuyo objetivo fundamental es desarrollar y mantener un producto software de interés para el cliente.

Business Process Management (BPM) es la estrategia de carácter técnico y empresarial para la mejora del desempeño de las organizaciones. La idea subyacente en BPM es que las organizaciones definan sus actividades en términos

de sus procesos de negocio, independientemente de si son para la prestación de servicios, el desarrollo o el mantenimiento de productos. La estrategia de BPM, desde el punto de vista de la gestión de los procesos software, abarca desde la definición explicita de los mismos, hasta la evaluación de los procesos desplegados sobre entornos reales, siguiendo así el ciclo de vida (véase la figura 3.1) propuesto por Weske [191].

A la hora de definir las cuestiones de investigación, hemos tomado como punto de partida las fases del ciclo de vida de BPM. Estas cuestiones conducen los pasos posteriores en el proceso de revisión, esto es, la búsqueda de la información, la extracción de datos, la clasificación y el análisis de los trabajos encontrados. A continuación, se describen las cuestiones planteadas, todas ellas encaminadas a estudiar las posibilidades de los modelos SPEM.

- *RQ1.* ¿Qué metodologías y procesos de Ingeniería del Software se han modelado con SPEM?
- *RQ2.* ¿Cuáles son los mecanismos o procedimientos para verificar y validar los modelos diseñados con SPEM?
- *RQ3.* ¿Cuáles son los beneficios que se obtienen al desplegar y ejecutar procesos software definidos en SPEM?
- *RQ4.* ¿Cuáles son las mejoras con respecto a la monitorización y evaluación de procesos software SPEM desplegados sobre entornos de ejecución?

Al responder a la cuestión *RQ1*, se puede medir el grado de aceptación del metamodelo como mecanismo para definir formalmente procesos software reconocidos. Con la cuestión *RQ2*, se pretende identificar las técnicas utilizadas para verificar y validar los procesos, una vez han sido diseñados. SPEM no sólo está orientado a la definición de procesos software, sino que también a su posterior despliegue en sistemas software. Por ello, la cuestión *RQ3* se focaliza en encontrar las iniciativas relativas a la ejecución de los procesos software. Finalmente, la cuestión *RQ4* pretende localizar cómo se puede monitorizar y mejorar los procesos software desplegados en entornos reales.

3.1.2. Estrategia de búsqueda

A continuación, se describe la estrategia de búsqueda de la bibliografía. Para ello, se seleccionaron una serie de bibliotecas digitales sobre las cuales realizar las búsquedas, en este caso: *Wiley On-line Library*, *World Scientific Net*, *IEEE Digital Library*, *Elsevier*, *Springer* y *ACM Digital Library*.

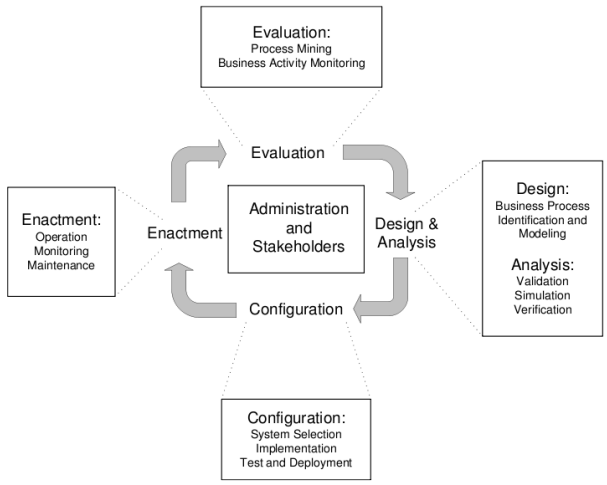

Figura 3.1: Ciclo de vida de los procesos de negocio

Existen multitud de términos relacionados con los procesos en Ingeniería del Software. Dado que el estudio de alcance es muy amplio, se optó por usar el término "*SPEM*" como la principal cadena de búsqueda en los diferentes motores y así localizar tantos artículos como sea posible. Sin embargo, debido a que este término tiene connotaciones en otras disciplinas, se añadió el término "*software process*" con el objetivo de reducir el ámbito de las búsquedas y reducir el volumen de resultados. Estos términos de búsqueda fueron posteriormente refinados para ajustarse a las características de los diferentes motores: modificando la sintaxis, estableciendo los campos sobre los que buscar, etc., pero siempre respetando la semántica de la búsqueda.

Con el objetivo de hacer este estudio lo más repetible y transparente posible se documentaron cada una de las búsquedas realizadas, registrando las fuentes usadas, los términos de búsqueda empleados y los resultados. Para cada uno de los resultados obtenidos se descargaron los documentos y los metadatos correspondientes, que luego fueron importados en una herramienta de gestión bibliográfica (basada en *BibTeX*). Algunos de los resultados obtenidos fueron rápidamente descartados por tratarse de índices de libros de investigación o de actas de congresos y también en aquellos casos donde el acceso al texto completo

no estaba disponible. Posteriormente, se refinaron y corrigieron los metadatos de los trabajos encontrados: eliminando registros duplicados, normalizando los nombres de los autores, publicaciones y palabras claves o *keywords*, y completando campos vacíos como el año de publicación o el editor.

3.1.3. Criterios de selección de estudios

A la hora de discriminar cuáles fueron los estudios primarios de interés para el estudio, se revisaron los títulos, resúmenes y palabras clave de los diferentes trabajos localizados. En ocasiones fue necesario realizar una revisión rápida de las conclusiones y del propio contenido de los artículos, dado que no siempre los resúmenes expresaban de forma clara las contribuciones fundamentales del trabajo.

La búsqueda se focalizó en localizar los trabajos relativos al diseño, configuración, ejecución y evaluación de procesos software definidos con SPEM. Para discernir entre los trabajos se establecieron una serie de criterios de exclusión:

- *Out of scope:* Trabajo publicado con anterioridad a Noviembre de 2002, fecha de publicación de la primera versión formal de SPEM.
- *Unsupported language:* Trabajo publicados en idioma diferente al inglés o al castellano.
- *Off-topic:* Trabajo no directamente relacionado con los tópicos de interés.
- *Duplicated:* Trabajo cuya contribución es similar o idéntica a otro trabajo ya incluido como estudio primario.

Con el primer criterio se consiguió descartar aquellos trabajos basados en versiones preliminares del lenguaje SPEM. El segundo criterio permitió eliminar los trabajos con menor repercusión, dado que los artículos no escritos en inglés tienen una visibilidad más limitada. El criterio *Off-topic* requirió mayor esfuerzo para distinguir si un determinado artículo es de interés para los tópicos de la investigación. De esta forma, se excluyeron las publicaciones que mencionaban SPEM simplemente en el trabajo futuro, o como mera notación visual para clarificar algún aspecto del trabajo.

Para descartar las publicaciones con idéntica o similar contribución, se llevó a cabo un proceso (no sistemático) de revisión de las publicaciones de cada uno de los autores con más de un artículo relevante. En aquellos casos donde encontramos publicaciones similares se seleccionó únicamente la más significativa. Para tomar esta decisión se tuvieron en cuenta los siguientes criterios: (i)

la importancia relativa de la publicación, por ejemplo, artículo en revista por delante de publicación en acta de congreso; (ii) el último artículo publicado en la evolución natural de la línea de investigación; (iii) mayor énfasis en el uso de SPEM y (iv) la extensión en número de páginas del propio trabajo.

3.1.4. Diseño del esquema de clasificación

Una vez se han seleccionado los estudios primarios de interés, es fundamental extraer información de los mismos, con el objetivo de intentar dar respuesta a las preguntas de investigación. Con este fin, se diseñó un formulario de extracción de datos en una hoja de cálculo sobre la cual se iban registrando todas las actividades de nuestro proceso. Además de los metadatos específicos y una breve descripción de cada una de las publicaciones, se registraron una serie de datos adicionales para cada uno de los estudios primarios que se describen a continuación.

Tipo de investigación

El tipo de investigación se refiere al enfoque utilizado por los autores de las distintas publicaciones. Se ha utilizado el esquema de clasificación propuesto en [192], dado que es el esquema recomendado para la realización de estudios de alcance en Ingeniería del Software [143]. En esta clasificación podemos distinguir entre:

- *Solution proposal:* En este tipo de publicación se propone una solución para un problema determinado. La solución puede ser totalmente novedosa o bien una extensión a una técnica ya existente. Los beneficios potenciales y la aplicabilidad de la solución son mostrados mediante un pequeño ejemplo o una adecuada línea de argumentación.

- *Validation research:* Se proponen soluciones noveles o que no han sido implementadas en la práctica. Estas soluciones se validan mediante experimentos, esto es, mediante trabajos desarrollados en laboratorio.

- *Evaluation research:* Se lleva a cabo una evaluación mediante su puesta en práctica, mostrando cuales son las consecuencias de su utilización en términos de beneficios y carencias.

- *Experience papers:* Trabajos donde se explica la experiencia personal del autor a la hora de poner algo en práctica.

- *Opinion papers:* Trabajos que expresan la opinión personal del autor con respecto a si una determinada técnica es adecuada o errónea o cómo deberían realizarse ciertas cosas. Este tipo de contribuciones no suelen basarse en trabajos relacionados ni en una metodología de investigación bien definida.

- *Philosophical papers:* En estos trabajos se exponen puntos de vista alternativos sobre algún ámbito del conocimiento, haciendo uso de taxonomías o marcos conceptuales.

Tipo de contribución

Un aspecto importante en el estudio de la literatura es identificar qué tipo de contribución aporta el estudio primario localizado. Sin embargo, es habitual la existencia de términos ambiguos y que los autores usan de manera no precisa. Por ello, vamos a describir los tipos de contribuciones sobre los que catalogaremos los artículos localizados. El esquema de clasificación utilizado es el siguiente:

- *Process:* Comprende aquellos trabajos cuya contribución principal viene en forma de proceso software, de contenido de método (*method content* en terminología SPEM) o de metodologías.

- *Model:* Comprende aquellos trabajos cuya contribución es en forma de una extensión al metamodelo de SPEM o bien diseñando uno nuevo basado en él.

- *Tool:* Utilizado en las publicaciones que presentan una herramienta software o una extensión a alguna ya existente.

- *Framework:* Aquí englobamos a los trabajos que contribuyen con una combinación de los tres elementos anteriores, esto es, al menos un proceso, un modelo y una herramienta.

- *Mapping:* Las publicaciones que describen un proceso de transformación *Model-to-Model* (M2M) o *Model-to-Text* (M2T) o bien indican las relaciones entre los elementos de modelos origen y destino se engloban bajo este criterio.

- *Technique:* Comprende aquellas publicaciones que describen un determinado procedimiento para llevar a cabo una actividad o tarea específica. Podría venir acompañada por una herramienta de soporte.

Hay que tener en cuenta que el título de la publicación no tiene por qué coincidir con algún tipo de contribución indicado anteriormente. Por ejemplo, en [110] la principal contribución es un nuevo modelo y no un proceso de desarrollo como podría parecer.

Versión del metamodelo

Este criterio indica la versión del lenguaje SPEM utilizada en las publicaciones.

- *SPEM 1.1:* Primera versión pública liberada por la OMG.
- *SPEM 2.0:* Versión actual del metamodelo.
- *SPEM 1.1 Extension:* Extensión a la primera versión del metamodelo.
- *SPEM 2.0 Extension:* Extensión a la versión actual del metamodelo.
- *SPEM 1.1 Devised:* Nuevo metamodelo diferente a SPEM 1.1, aunque ideado a partir de él.
- *SPEM 2.0 Devised:* Nuevo metamodelo ideado en la versión 2.0 de SPEM.
- *UMA:* Adaptación del metamodelo SPEM 1.1 por parte de la comunidad *Eclipse*.

Ámbito de la Investigación

Además de los datos anteriores, es importante recolectar más información que nos permita conocer con mayor exactitud el ámbito de las diferentes contribuciones. Inicialmente, se plantearon las fases del ciclo de vida de BPM como esquema de clasificación. Sin embargo, estas fases fueron demasiado genéricas por lo que se necesitó una granularidad menor para los datos extraídos. Para resolver esto, realizamos una fase posterior de *keywording* o análisis de palabras clave. De esta forma, el esquema de clasificación fue evolucionando (creando nuevas categorías y subcategorías), al mismo tiempo que se fue llevando a cabo la extracción de datos de los artículos. El esquema final resultante resultó en:

- *Process modeling:* Modelado de metodologías y procesos en Ingeniería del Software.
- *Process adaptability:* Adaptación o *tailoring* de procesos software.

- *Process verification&validation:* Verificación y validación de los modelos de procesos software.
- *Process configuration&enactment:* Configuración y despliegue para la ejecución de los procesos software sobre entornos de soporte.
- *Process evaluation:* Evaluación de los procesos mediante la recolección de datos de los procesos en ejecución.

De esta forma, el esquema original basado en las fases del ciclo de vida BPM fue transformado en un grupo estructurado de actividades en el área de la gestión de procesos software, acordes a la literatura encontrada.

3.1.5. Visualización y análisis de datos

Con posterioridad a la categorización de las publicaciones según los criterios anteriores, se llevó a cabo una etapa de análisis donde los datos extraídos fueron agregados y resumidos para dar respuesta a las preguntas de investigación expuestas. El análisis de los resultados se focalizó en estudiar el número de publicaciones por cada categoría con el fin de exponer cómo de poblada se encuentra una determinada área de interés. Esta información es posteriormente resumida en forma de tablas o diagramas. Otro método útil para presentar la información y utilizado en este estudio es la combinación de diferentes categorías (por ejemplo, ámbito de la investigación vs. tipo de contribución) representándolas en un gráfico de burbujas.

3.2. Resultados

A continuación, se recogen los resultados del estudio de alcance desarrollado según el procedimiento descrito anteriormente. Los estudios primarios seleccionados están disponibles en un grupo visible públicamente en la plataforma online para investigadores *Mendeley*[1].

3.2.1. Selección de estudios

En la tabla 3.1 se presentan las búsquedas realizadas en las bibliotecas digitales más importantes en Ingeniería del Software, los términos empleados, el ámbito de búsqueda, el número de trabajos encontrados y el número de estudios primarios

[1] http://www.mendeley.com/groups/938811

Tabla 3.1: Búsquedas realizadas y publicaciones seleccionadas

Fuente de Datos	Términos	Ámbito de búsqueda	Rdos.	Sel.
World Scientific Net	*spem*	*Title and Keywords*	2	2
Wiley Online Library	*"software process" AND spem*	*All Fields (Including full text)*	10	2
IEEE Digital Library	*"software process" AND spem*	*Full Text and Metadata*	132	41
Elsevier (ScienceDirect)	*"software process" AND spem*	*Full Text*	39	13
Springer	*"software process" AND spem*	*Full Text*	145	41
ACM Digital Library	*"software process" AND spem*	*Full Text, Title, Abstract and Review*	45	16

seleccionados después de aplicar los criterios de selección y exclusión. En todos los casos, se utilizaron los formularios de búsqueda avanzada que ofrecen los motores de búsqueda.

Se obtuvieron un total de 373 publicaciones. Todas ellas fueron inspeccionadas para comprobar los criterios de inclusión y exclusión. El resultado de esta selección puede observarse en la tabla 3.2. Finalmente, fueron 115 los estudios primarios seleccionados para realizar su análisis, aproximadamente un 30 % de los estudios encontrados. En el diagrama de la figura 3.2, se muestra la distribución de los estudios primarios a lo largo de los últimos años. Los trabajos seleccionados proceden de diferentes fuentes, de modo que nos ofrece una buena idea del amplio rango y multidisciplinar de las áreas donde se utiliza SPEM.

Tabla 3.2: Resultados de la selección de estudios

Criterio	Publicaciones	Frecuencia
Off-topic	194	52,01 %
Duplicated	51	13,67 %
Out of scope	8	2,14 %
Unsupported language	5	1,34 %
Included	115	30,83 %
Total	373	100,00 %

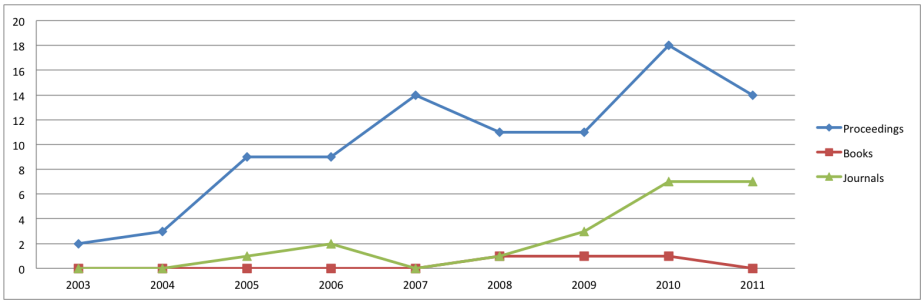

Figura 3.2: Distribución anual de los estudios primarios

Las principales fuentes de publicaciones, según la cantidad de estudios localizados, son las conferencias *International Conference on Software Process*, *Asia-Pacific Software Engineering Conference*, *ACM Symposium on Applied Computing* y la revista *Information and Software Technology*. En la tabla 3.3 se incluyen las conferencias y las revistas donde se han publicado más de un estudio primario. La preparación de esta distribución requirió un proceso adicional de normalización de los metadatos recolectados. En primer lugar, se homogenizaron los nombres de las conferencias, eliminando los prefijos '*Proceedings of*', los acrónimos, los números de edición y los años de celebración. Además, se localizaron las actas de ciertas conferencias que en ocasiones se publican con nombres diferentes, como sucede en el caso de *Springer LNCS*.

3.2.2. Extracción de datos

En los estudios analizados se pudo comprobar que no sólo se utilizan las dos versiones estándar de SPEM, sino que además se proponen bastantes extensiones o mejoras a estos metamodelos. En la figura 3.3 podemos comprobar la frecuencia de uso de los metamodelos.

El diagrama de burbujas de la figura 3.4 muestra la distribución del ámbito de investigación de los estudios primarios según el tipo de contribución que ofrecen y según el tipo de investigación realizada. En este diagrama podemos observar cómo la definición de procesos y, en menor extensión, el desarrollo de *frameworks* son las contribuciones más frecuentes. Desde la perspectiva del tipo de investigación, la mayor parte de las publicaciones son simples propuestas de solución y luego seguidas de soluciones con validación.

Tabla 3.3: Distribución de las fuentes de publicación

Fuente de publicación	Publicaciones
Conferencias	
Int. Conf. on Software Process	6
Asia-Pacific Software Engineering Conference	6
ACM symposium on Applied Computing	5
Workshop on Software engineering for sensor network applications	4
Eur. Conf. on Software Architecture	3
Int. Conf. on Product-Focused Software Process Improvement	3
Int. Conf. on Computer Science and Information Technology	2
Eur. Conf. on Model Driven Architecture: Foundations and Applications	2
Int. Conf. on Software Engineering Advances	2
Int. Conf. on Advanced Communication Technology	2
Int. Conf. on Web Engineering	2
Int. Conf. on Advanced Information Systems Engineering	2
Int. Workshop on Agent-Oriented Software Engineering	2
Int. Conf. on Information Technology: New Generations	2
Int. Conf. on Soft. Eng., Research, Management and Applications	2
Revistas	
Information and Software Technology	4
Journal of Software Maintenance and Evolution: Research and Practice	2
Journal of Software Engineering and Knowledge Engineering	2
Journal of Systems and Software	2

3.2.3. Clasificación de los estudios

En los siguientes apartados se presentan los resultados del estudio de alcance para cada una de las áreas de investigación. Adicionalmente, en el anexo A se incluye la clasificación completa de los estudios primarios.

Modelado de procesos

De todas las posibles aplicaciones del lenguaje SPEM y de sus derivados, el modelado de procesos es, sin duda, la más extendida. Existe un amplio rango de procesos que son modelados con este lenguaje, desde metodologías para el desarrollo de sistemas en tiempo real, a metodologías especificas para el desarrollo de sistemas relativos a la salud, por ejemplo. Todos los estudios fueron analizados y clasificados en cinco áreas principales no disjuntas. Los enfoques generalistas para el desarrollo de software y aquellos específicos para ciertas tecnologías son las áreas más pobladas en cuanto al número de contribuciones significativas. En

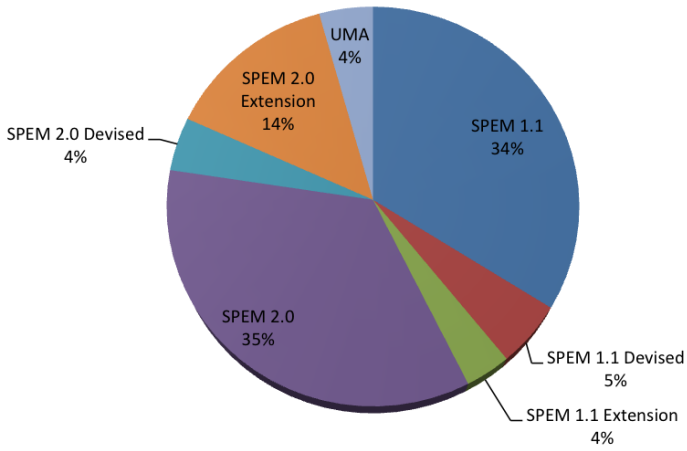

Figura 3.3: Versiones del metamodelo empleadas en las publicaciones

la tabla 3.4 se muestra el número de publicaciones encontradas para cada tipo de metodología o proceso modelado con SPEM.

Con respecto a los enfoques genéricos de desarrollo, encontramos muchas contribuciones relativas al diseño de metodologías para líneas de productos software, para el desarrollo ágil y para los basados en UP. En menor medida, SPEM también se utiliza para representar procesos de desarrollo dirigidos por modelos y aquellos orientados al desarrollo de software libre. Con respecto a los enfoques para tecnologías específicas, podemos encontrar métodos para el desarrollo de sistemas multiagente o MAS, de interfaces de usuario, de componentes web, de almacenes de datos, de aplicaciones móviles, de ficheros XML y para la reutilización de *frameworks* orientados a objetos.

Además, existen diversos trabajos que utilizan SPEM para modelar aspectos de la gestión de cambios en el software, ingeniería de requisitos y calidad de productos y procesos software (CMMI), entre otros. Por otra parte, existe un importante número de trabajos relativos al uso de SPEM para modelar sistemas con restricciones importantes en cuanto a tiempos de respuesta, seguridad o rendimiento. Asimismo, se pueden citar estudios aislados donde se proponen procesos específicos para el desarrollo de sistemas de salud o de automoción, entre otros.

Tabla 3.4: Tipos de procesos modelados con SPEM

Categoría	Publicaciones
Enfoques generalistas	
Software Product Lines (SPL)	8
Agile Software Development (ASD)	7
Unified Process (UP)	7
Model-Driven Development (MDD)	5
Open Source Software Development (OSSD)	2
Enfoques específicos de tecnología	
Multi-Agent Systems (MAS) Development	13
User Interface (UI) Development	4
Web Services (WS) and Applications Development	3
Context-aware Systems (CaS) Development	2
Data Warehouse (DW) Development	2
Mobile Application Development (MAD)	1
Object-oriented (OO) frameworks	1
Development with eXtensible Markup Language (XML)	1
Iniciativas para la mejora de procesos y productos software	
Capability Maturity Model Integration (CMMI)	5
Change Management (CM)	5
Multi-Perspective Process Modeling (MPM)	4
Software Process Assessment (SPA)	2
Requirements Engineering (RE)	2
Product Quality Improvement (PQI)	1
Sistemas con restricciones no funcionales específicas	
Dev. of Embedded Real-Time Systems (ERT)	7
Dev. of Secure Systems (SS)	3
Dev. of High Performance Systems (HPS)	2
Enfoques para dominios específicos	
Dev. of Customer Relationship Management Systems (CRM)	1
Dev. of Software Ecosystems (SOFTECO)	1
Dev. of Health Care Systems (HCS)	1
Dev. of Automotive Systems (AS)	1
Dev. of Learning Resources (LR)	1

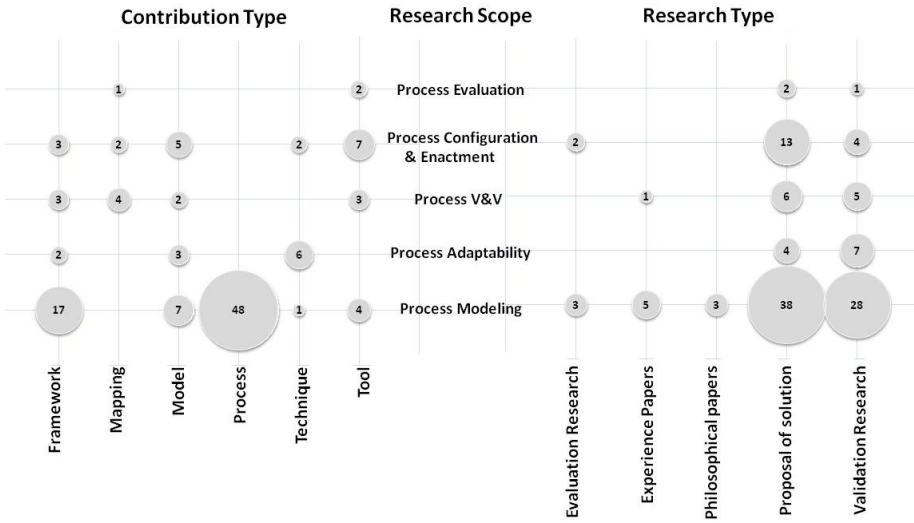

Figura 3.4: Ámbito de la investigación distribuido sobre el tipo de investigación y tipo de contribución

En la tabla A.1 se ofrece la lista completa de trabajos relacionados junto con el tipo de proceso modelado. Hay que destacar que los tipos no son mutuamente exclusivos y que por tanto, algunos de los estudios cubren más de un tipo.

Adaptabilidad de procesos

Debido a la naturaleza variable que tienen los propios procesos software, es complicado encontrar dos organizaciones que utilicen una metodología idéntica para la ejecución de sus proyectos, ya que en Ingeniería del Software nunca existen dos proyectos iguales. Por ello, se está dando gran importancia a la idea de modelar la flexibilidad de los procesos software, mediante la SME y las *Software Product Lines* (SPL). Para ello, se diseñan fragmentos de métodos en lugar de metodologías completas, de forma que éstas puedan ser adaptadas o personalizadas según las necesidades particulares de cada proyecto.

En la tabla A.2 se recogen los estudios dedicados a la adaptabilidad de procesos. En estos trabajos se presentan contribuciones relativas al modelado de la

parte común y la variabilidad de los procesos, al razonamiento basado en casos, al uso de la orientación a aspectos durante el modelado, a la adaptación mediante la transformación de modelos, a la reutilización basada en arquitecturas software y al control de versiones de los propios procesos.

Verificación y validación de procesos

Los modelos de procesos software deben ser analizados para detectar posibles inconvenientes. Mediante la utilización de modelos como base para los procesos software, se consigue facilitar la reutilización de técnicas estándar para la verificación y validación de modelos.

En la revisión de la literatura se encontraron diversos trabajos (véase la tabla A.3) que tratan ciertos aspectos de la verificación y validación de modelos. En las publicaciones localizadas se describen técnicas formales de verificación basadas en Redes de Petri, *Object Constraint Language* (OCL) y ontologías. También se presentan técnicas de análisis visual sobre la efectividad y usabilidad de los modelos, así como métricas para el calculo de la adherencia con respecto a un determinado *framework* de mejora, como CMMI. Asimismo, existen otros trabajos interesantes relativos a la validación de los modelos SPEM mediante técnicas basadas en la transformación de éstos en modelos interpretables en entornos de simulación analítica o de aprendizaje, como *DEVS-Hybrid* [162] o *SimSE* [130].

Configuración y despliegue de procesos

Una vez los modelos de procesos han sido verificados y validados, deben implementarse en entornos software de tal modo que la ejecución de los proyectos se gestione como instancias de los procesos. Las publicaciones relativas a este área se muestran en la tabla A.4.

Los procesos software, como cualquier otro tipo de proceso de negocio, pueden ser desplegados en sistemas de gestión de procesos de propósito general, mediante la utilización de lenguajes basados en flujos de trabajos o *workflows*. En la literatura, se encontraron diversos trabajos que proponen transformaciones de modelos SPEM en modelos de *workflows* basados en UML, XPDL, JPDL o *Business Process Execution Language* (BPEL). Estas transformaciones están diseñadas para lograr la orquestación automática de servicios web en plataformas de integración de aplicaciones. Asimismo, se localizaron diversos trabajos en donde se proponen mejoras al metamodelo para superar ciertas carencias con respecto a la planificación y ejecución de los procesos.

Resulta interesante destacar que algunas herramientas del mercado, como *IRIS Process Author*, permiten generar automáticamente *workitems* para la plataforma *Visual Studio ALM*[2], o EPF, que permite generar plantillas de planificación *Gantt* para *Microsoft Project*[3] a partir de un modelo de procesos SPEM.

Además, se encontraron diferentes propuestas que ofrecen soporte parcial a la automatización de los procesos software. Podemos citar al entorno de modelado y despliegue basado en *Model-Driven Architecture* (MDA) propuesto en [117], a la herramienta que incrusta la documentación de los procesos software dentro de los propios entornos de desarrollo [88] y a la propuesta de generación semi-automática de entornos de desarrollo extensibles sobre la plataforma *Eclipse* [30].

Evaluación de procesos

Los procesos de negocio pueden ser evaluados utilizando herramientas de monitorización en tiempo real o de análisis post-mortem, habitualmente integradas en los sistemas BPM. Desde la perspectiva de los procesos software, se encontraron algunas propuestas (tabla A.5) relativas a: (i) la medición de procesos a partir de la recolección de datos de herramientas de soporte al desarrollo; (ii) la detección de inconsistencias entre la definición de los procesos y los datos recogidos de los proyectos, haciendo uso de tecnologías de la web semántica y (iii) a la utilización de técnicas de *model relaxing* y *model changing* para adaptar dinámicamente los modelos de procesos.

3.3. Discusión

En esta sección se resumen los usos y aplicaciones del lenguaje de modelado, las extensiones que diferentes autores han propuesto al estándar y el grado de aceptación del mismo. Finalmente, se describen las posibles amenazas a la validez del estudio de la literatura realizado.

3.3.1. Usos y aplicaciones de SPEM

El principal objetivo de SPEM, como cualquier otro lenguaje de definición de procesos, es proporcionar los elementos necesarios para representar procesos

[2] http://www.microsoft.com/visualstudio/esn/alm
[3] http://office.microsoft.com/en-us/project/

software de forma estándar. En términos generales, podemos afirmar que la gestión de procesos software modelados con SPEM es un área que hoy día requiere mayor investigación. La tendencia ascendente en el número de contribuciones científicas observadas confirma que se trata de un área de investigación en auge.

Actualmente, la mayor parte de los trabajos se centran principalmente en la definición de metodologías y procesos, así como en la propuesta de nuevos metamodelos y herramientas de soporte. Un aspecto importante a destacar es la gran cantidad de propuestas de solución existentes comparadas con la escasez de trabajos que presentan evaluaciones formales. Hay una carencia evidente de experiencias reales que hagan uso de las contribuciones propuestas, así como de evaluaciones realizadas en la industria.

A continuación, se presentan las respuestas para cada una de las cuestiones de investigación planteadas en este capítulo y refutadas por los resultados de la revisión de la literatura realizada.

- *RQ1. ¿Qué metodologías y procesos de Ingeniería del Software se han modelado con SPEM?* A partir de lo encontrado en la literatura, se puede afirmar que los procesos software más implementados con SPEM son los relativos al desarrollo de sistemas multiagente y en menor medida, al diseño de líneas de productos software, al desarrollo ágil de software y a aquellos basados en UP. El gran éxito de SPEM en el modelado de procesos para sistemas multiagente se debe fundamentalmente a que fue recomendado como formalismo para la representación de este tipo de procesos en el seno del comité técnico de la *Foundation for Intelligent Physical Agents* (FIPA) [35].

- *RQ2. ¿Cuáles son los mecanismos o procedimientos para verificar y validar los modelos diseñados con SPEM?* En la literatura se encontraron varias publicaciones describiendo técnicas muy diversas para la verificación y validación de modelos SPEM, entre otras, la utilización de restricciones en OCL, el cálculo de métricas y la simulación mediante entornos específicos.

- *RQ3. ¿Cuáles son los beneficios que se obtienen al desplegar y ejecutar procesos software definidos en SPEM?* Los trabajos encontrados a este respecto tratan mayoritariamente con la transformación de los modelos de procesos en modelos de *workflow* para, idealmente, ser ejecutados en los sistemas de gestión de procesos de propósito general, con el objetivo de explotar las ventajas que ofrecen estos sistemas.

- *RQ4. ¿Cuáles son las mejoras con respecto a la monitorización y evaluación de procesos software SPEM desplegados sobre entornos de ejecución?*

El área menos poblada en cuanto al número de artículos es la relativa a la evaluación y monitorización de los procesos en ejecución. Los entornos específicos dedicados a los procesos software se encuentran en fases muy preliminares, por lo que resulta complejo recolectar datos de proyectos reales, a diferencia de los sistemas genéricos BPM que ya disponen un alto nivel de madurez y un importante calado en la industria.

3.3.2. Extensiones al lenguaje

A pesar del visible incremento en el uso de SPEM y los beneficios obtenidos con la ultima versión, parece que la especificación es aún incompleta, como evidencia el importante número de trabajos que proponen extensiones al estándar o nuevos lenguajes derivados del mismo. El desarrollo de extensiones o derivados de SPEM ha sido una práctica común en los investigadores, tanto en la primera versión del metamodelo como en la más reciente. A continuación, se describen las principales extensiones encontradas.

Un importante conjunto de extensiones fueron diseñadas para modelar aspectos concretos de algunos procesos específicos que no podían ser modelados completamente con la especificación actual. Por ejemplo, los procesos basados en prácticas agiles [100], en metodologías orientadas a agentes [170] y en métodos para la construcción de interfaces de usuario [176]. También se han propuesto dos extensiones para incluir ciertos aspectos de CMMI [120, 109].

Por otra parte, se desarrollaron un conjunto de extensiones [6, 121, 122, 190] con el objetivo de mejorar la flexibilidad de los procesos utilizando nuevos mecanismos de variabilidad. Con estos mecanismos, se pretende que los ingenieros de procesos puedan definir y publicar modelos de procesos software expresando la parte común y la variabilidad en los flujos de trabajo. Estas extensiones habilitan la construcción de líneas de procesos software o SPL, y la derivación de procesos específicos para proyectos concretos (*tailoring*).

Existen otras extensiones a SPEM relativas al modelado de procesos software, pero desde diferentes perspectivas. Por ejemplo, en [91] el modelado de procesos se realiza en base a las dimensiones de tecnología, riesgos, organización y personas. En [110] las dimensiones son personas, procesos y tecnología. En [174] el modelado se centra en la autoría de los artefactos software. Otra extensión a destacar es la propuesta en [153] dedicada a la representación explícita del conocimiento de las organizaciones.

SPEM 2.0 se definió como un metamodelo basado en MOF y como un perfil para UML. La mayor parte de los autores que propusieron mejoras a SPEM se basan en el metamodelo, mientras que sólo una pequeña parte confiaron en

la notación UML. El enfoque más utilizado para extender SPEM fue la creación de nuevos paquetes que especializaban conceptos y relaciones existentes en el metamodelo, o bien, mediante la creación de nuevos elementos enriquecidos con la semántica necesaria para asegurar su corrección y consistencia. Sin embargo, muy pocos trabajos emplearon OCL como mecanismo para expresar las restricciones semánticas.

El soporte de la versión 2.0 de SPEM para la ejecución de procesos tiene todavía mucho margen de mejora. SPEM no puede ser directamente interpretado por ningún sistema ni puede ser mapeado directamente en otro lenguaje ejecutable. En [15, 52, 173] se analizan los detalles de los elementos relacionados con el modelado de comportamiento y ejecución dentro del metamodelo de SPEM. En estos estudios, los autores proponen un conjunto de extensiones que incluyen nuevos conceptos y la semántica asociada, con el objetivo de lograr la ejecución automatizada del proceso de desarrollo. También se proponen mejoras específicas para la planificación de actividades y la asignación de recursos.

En [17] se presenta una completa evaluación de los enfoques basados en UML (incluyendo SPEM) con respecto a los requisitos deseables que los lenguajes de modelado de procesos software deberían satisfacer. En este trabajo, se destaca la riqueza semántica del lenguaje para definir todos los elementos implicados en el proceso y su modularidad. Sin embargo, se resaltan otras desventajas de SPEM como el poco soporte a la ejecución de los procesos y la carencia de control reactivo para el manejo de eventos externos y fallos durante las actividades del proceso.

Todas las extensiones anteriores están muy acopladas al diseño del metamodelo. Sin embargo, podemos citar una propuesta muy interesante [122] que extiende las metaclases de SPEM pero de forma no intrusiva, utilizando para ello el patrón de diseño *Decorator* [189]. Este patrón permite añadir nuevas responsabilidades a los objetos de forma dinámica, extendiendo de esta manera su funcionalidad en tiempo de ejecución.

Todas las extensiones descritas anteriormente tienen una clara motivación, la de mejorar ciertas carencias del lenguaje SPEM. Sin embargo, ninguna de las extensiones propuestas fueron elaboradas como resultado de un debate público o consenso entre grupos de expertos. En Internet podemos encontrar varias definiciones de procesos software modeladas con SPEM, pero no sucede lo mismo con las extensiones, las cuales sólo aparecen descritas en las propias publicaciones. Esta es una de las razones por la cual ninguna de las extensiones diseñadas están soportadas por las herramientas líderes en modelado SPEM.

3.3.3. Aceptación del estándar

SPEM es un estándar de la industria que pretende facilitar el entendimiento y la comunicación de los procesos software y así potenciar su reutilización y mejora. La versión 2.0 de SPEM añade nuevas características sobre la versión inicial del lenguaje, dirigidas a separar la definición del método de su aplicación real en un proceso, gestionar familias enteras de procesos y diferentes modelos de ciclo de vida, mejorar la variabilidad mediante mecanismos de extensión vía *plugins* y reutilizar patrones y componentes de procesos.

La definición de procesos software haciendo uso de un lenguaje de modelado computable, como SPEM, ofrece diferentes beneficios para todas las partes implicadas. Sin embargo, el lenguaje no ha conseguido suficiente nivel de aceptación en la industria y hasta la fecha, sólo se utiliza mayoritariamente en ámbitos académicos y de investigación. Pocos trabajos relativos a la evaluación y utilización real de las propuestas se encontraron durante la revisión de la literatura.

El diseño de procesos con SPEM no es una tarea sencilla debido a la complejidad del propio lenguaje, que requiere un profundo entendimiento de la especificación. Esto y la ausencia de mecanismos eficaces para la ejecución de procesos origina un retorno de la inversión demasiado bajo como para que modelar procesos con SPEM sea una actividad de interés para la mayoría de las empresas.

Son varias las condiciones que se deberían satisfacer para consolidar SPEM. Por un lado, se requiere refinar y mejorar la versión actual del estándar incorporando las extensiones propuestas por los diferentes autores y, por otro, disponer de un compromiso importante por parte de los desarrolladores de herramientas para mejorar la etapa de modelado y en especial la etapa de despliegue de los procesos sobre entornos automatizados. En cierta medida, la gestión de procesos software basada en un enfoque dirigido por modelos está teniendo los mismos problemas que la metodología MDA para el desarrollo de software, esto es la necesidad de grandes esfuerzos en aprendizaje y de entrenamiento para poder utilizarlo de forma efectiva [102].

3.3.4. Amenazas a la validez

En los estudios sistemáticos de la literatura es esencial identificar los problemas potenciales de sesgo o validación que se puedan derivar durante el diseño y desarrollo de la revisión. De esta forma, se pretende dotar de un alto grado de confianza a los resultados que se han presentado.

Para asegurar la *validez de construcción* del estudio se utilizaron unas pautas

que son ampliamente reconocidas en Ingeniería del Software. Adicionalmente, para intentar lograr el máximo número de trabajos de interés, que nos ayuden a responder a las preguntas de investigación, se han utilizado términos de búsqueda generales y las consultas se lanzaron sobre las bases de datos electrónicas de investigación más relevantes y extendidas. En ocasiones, aparecían ambigüedades con respecto al tipo de investigación propuesto en los artículos, así como de los términos específicos empleados. Dado que el desarrollo del estudio de alcance no incluía un estudio en profundidad de los estudios primarios, es posible que algunos juicios de error se hayan cometido a la hora de clasificar los artículos en las categorías especificadas. Sin embargo, se ha intentado ser lo mas cuidadoso posible a la hora de llevar a cabo esta tarea.

Además de lo anterior, es fundamental evaluar la validez interna y externa del estudio. La *validez interna* se refiere a la medida por la cual el diseño del método de estudio y su ejecución intenta prevenir la obtención de resultados sesgados. En nuestro caso, la validez interna queda garantizada ya que a la hora de realizar el análisis de los resultados, únicamente utilizamos técnicas sencillas de conteo y de visualización para representar los datos extraídos. De esta forma, describir los hallazgos de la investigación ha sido una labor relativamente sencilla. La *validez externa* mide si los efectos observados en el estudio son aplicables fuera del presente estudio. Dado que en este trabajo no se han propuesto generalizaciones, afirmaciones o proyecciones, la validez externa queda también asegurada.

Otro aspecto clave para asegurar la calidad de la revisión de la literatura es su *fiabilidad*. La fiabilidad se intenta conseguir facilitando a otros investigadores la posibilidad de repetir el estudio, de forma que obtengan resultados idénticos o similares a los presentados. Para ello, tanto el procedimiento seguido como los resultados detallados se han publicado en el sitio web de apoyo a esta tesis y todas las referencias analizadas están agrupadas en una popular plataforma online de investigación.

Parte III

Framework para el despliegue y evaluación de procesos software

Capítulo 4

Planteamiento del problema

En este capítulo se resumen los problemas principales que motivan la presente investigación, así como las hipótesis que dirigen las contribuciones presentadas en posteriores capítulos.

4.1. Falta de alineamiento entre los procesos y las herramientas

Los procesos de negocio, y por ende de software, siguen un ciclo de mejora continua, comenzando por el diseño de los mismos con algún lenguaje estándar como BPMN, la verificación y validación, la configuración y despliegue para la ejecución (*enactment*), y la evaluación. La etapa de configuración de los procesos de negocio consiste en desplegar los modelos de procesos en algún motor BPM. Además de esto, es necesario definir los mecanismos de interacción con el usuario final y la integración con sistemas heredados, habitualmente mediante la implementación de módulos o extensiones al sistema.

A diferencia de la gestión de procesos de negocio controlada por los sistemas BPM, en Ingeniería del Software no disponemos de suites integrales para la definición, configuración, ejecución y evaluación de los procesos software. Sin embargo, en los últimos años y gracias en parte al auge del movimiento *open source*, han surgido numerosas herramientas de soporte a la gestión y producción del software [27]. Estas herramientas pretenden agilizar y coordinar los equipos de trabajo, así como mejorar la calidad de los productos generados. En la tabla 4.1 podemos ver algunos tipos y ejemplos de herramientas comunes.

Tabla 4.1: Tipos y ejemplos de herramientas de soporte

Sistema	Descripción	Ejemplo
Monitorización de proyectos	Herramientas para la gestión de tareas, incidencias, versiones, etc.	*Redmine*[1]
Edición colaborativa	Herramientas wiki para la edición colaborativa de documentos en la web.	*MediaWiki*[2]
Modelado de sistemas software	Herramientas para el modelado de sistemas mediante lenguajes como UML, BPMN, etc.	*Visual Paradigm for UML*[3]
Gestión de listas de correo	Herramientas para la distribución de correos y administración de suscripciones a listas.	*Mailman*[4]
Gestión de foros	Herramientas para la publicación de opiniones y discusiones.	*phpBB*[5]
Control de código fuente	Herramientas para la gestión de revisiones de código fuente.	*Subversion*[6]
Gestión de pruebas	Herramientas para administrar el conjunto de pruebas sobre el software.	*TestLink*[7]
Gestión de componentes	Herramientas para la administración y publicación de componentes software.	*Nexus*[8]
Integración continua	Herramientas para la ejecución de tareas de construcción y despliegue de software.	*Jenkins*[9]
Gestión documental	Herramientas para la administración, versionado y flujos de revisión sobre documentos.	*Alfresco*[10]
Gestión de requisitos	Herramientas para la gestión de requisitos, gestión de cambios y su trazabilidad.	*IBM Rational DOORS*[11]
Gestión de contenidos	Herramientas para la gestión de noticias y contenidos web.	*WordPress*[12]
Gestión de aprendizaje	Herramientas para el diseño y ejecución de cursos de aprendizaje.	*Moodle*[13]

En paralelo al desarrollo de estos tipos de herramientas han ido apareciendo diversas plataformas para fomentar la cooperación entre desarrolladores en la gestión y difusión de software, así como para ofrecer soporte al usuario final. Estas plataformas se denominan forjas de software. Ejemplos de estos sistemas son *SourceForge*[14] , *GoogleCode*[15], *GitHub*[16] o *Assembla*[17], entre otros. Habitualmente, las forjas podemos encontrarlas como servicios en la web, *Software as a Service* (SAAS). En estos entornos se albergan múltiples proyectos de soft-

[14] http://sourceforge.net/
[15] https://code.google.com/intl/es/
[16] https://github.com/
[17] https://www.assembla.com/

ware, en los que los desarrolladores han de registrarse para poder contribuir. Las forjas de software suelen proporcionar algunas de las características propias de los tipos de herramientas identificadas anteriormente, como, por ejemplo, la monitorización de proyectos, la edición colaborativa o el control de código fuente.

Las forjas de software, a su vez, están evolucionando hacia el concepto de plataforma *Application Lifecycle Management* (ALM). Estas plataformas están diseñadas para integrar y coordinar las herramientas de ingeniería y gestión del software, con el objetivo de cubrir todas o gran parte de las actividades del ciclo de vida del software. Actualmente, la mayoría de las plataformas ALM existentes son propietarias y no libres, por lo que su uso aún no ha sido mayoritariamente adoptado.

Tal diversidad de herramientas de soporte origina que, al comienzo de cada nuevo proyecto, haya que dedicar esfuerzos considerables para la adaptación de las mismas a las necesidades concretas del proyecto y de la metodología implantada en la organización. Además, debido a la lenta aceptación del estándar SPEM, las herramientas de soporte, forjas o plataformas ALM no suelen incorporar mecanismos para vincular las definiciones explícitas (modelos) de los procesos [160]. Esto suele derivar en una falta importante de consistencia entre la definición de los procesos y luego la ejecución real de los proyectos.

Hipótesis 1
Las inconsistencias entre la definición de los procesos y la ejecución de los proyectos podrían minimizarse, en parte, mediante la personalización y adaptación de las herramientas de soporte y la creación de plantillas específicas para las mismas.

Sin embargo, realizar esta labor de forma manual es bastante compleja y costosa en tiempo, al no disponer de mecanismos para su automatización.

4.2. Complejidad en la evaluación de procesos software

En el ciclo de mejora continua de la calidad, la evaluación de los procesos es fundamental. Por ello, para poder aplicar mecanismos de mejora es necesario medir y analizar los errores, carencias o desviaciones en la ejecución real de los procesos. El análisis de métricas e indicadores permite mejorar la gestión de los

procesos software, dotando de medios para predecir y controlar la ejecución de los proyectos y para evaluar la calidad de los productos generados [51]. En la literatura podemos hallar citas importantes acerca de la medición en el software: "No se puede controlar lo que no se puede medir" [44] o "no se puede predecir lo que no se puede medir" [58].

El establecimiento de un plan de medición automatizado utilizando los sistemas de gestión de procesos de negocio suele ser una misión relativamente sencilla de implementar, debido a que estos sistemas suelen incluir herramientas de monitorización de métricas en tiempo real y motores de análisis post-mortem. En los procesos software, donde las plataformas integradas ALM no son ampliamente utilizadas, la medición resulta ser una actividad mucho más compleja.

En la sección anterior se citaron algunos tipos de herramientas que ofrecen soporte al proceso software y unos ejemplos concretos. Estos sistemas son las principales fuentes de información de los procesos software, ya que registran todos los hechos producidos durante el transcurso de las actividades del ciclo de vida. Algunos de estos sistemas, especialmente los dedicados al control y monitorización de proyectos, suelen incluir ciertas facilidades para la obtención de métricas relativas al número de errores, total de horas incurridas, etc. Sin embargo, no ocurre lo mismo con el resto de herramientas. Con el objetivo de superar esas limitaciones, alrededor de estas herramientas (fuentes primarias) se han ido construyendo otras (fuentes secundarias), que extraen y agregan los datos de los sistemas anteriores y presentan métricas en forma de tablas y/o gráficas. En la tabla 4.2 podemos ver algunos de estos ejemplos.

Tabla 4.2: Herramientas de extracción y agregación de datos

Sistema	Ejemplo
Análisis de sistemas de seguimiento de errores	*Bicho*[18]
Análisis de contribuciones en entornos wiki	*StatMediaWiki*[19]
Análisis de listas de correo	*MailListStats*[20]
Análisis de repositorios de código fuente	*CVSAnaly*[21]
Análisis de calidad de código	*SonarQube*[22]
Análisis de actividad de estudiantes	*Gismo*[23]

Además del análisis de métricas e indicadores, existen otras actividades de control muy importantes en Ingeniería del Software, como las revisiones técnicas. Estas actividades suelen ser bastante repetitivas y requieren de una asignación

importante de personal para realizarlas, ya que son actividades manuales. Las revisiones suelen realizarse en ciertos momentos a lo largo del ciclo de vida del software, como el fin de determinadas fases, actividades, iteraciones (en ciclos de vida evolutivos) o justo antes de entregas al cliente. Durante las revisiones se buscan evidencias del correcto o incorrecto uso de la metodología de la organización y de prácticas de Ingeniería del Software, empleando habitualmente listas de revisión o *checklists* definidas para tal fin. Estas listas suelen incluir la comprobación de la finalización de actividades de producción y de gestión, la correcta generación de los productos de trabajo y entregables, etc.

Aunque el análisis de indicadores o métricas y las revisiones de software son actividades fundamentales para la mejora de la calidad del software, suele ser habitual que no se le dediquen los esfuerzos y recursos suficientes. Esto se debe a que tradicionalmente las actividades de calidad se consideran que son poco productivas, en el sentido de que no conducen directamente a generar nuevo software. Por ello, es necesario investigar en mecanismos que permitan automatizar esta clase de actividades.

Hipótesis 2
Conseguir una visión global y uniforme de la información gestionada por las herramientas de soporte permitiría automatizar la recogida de métricas y la evaluación de la calidad en los procesos software.

Las herramientas de soporte a la gestión o la producción de software disponen de una gran cantidad de información que puede ser utilizada con el propósito de evaluar los procesos software. Sin embargo, la falta de sincronía, en términos de la interoperabilidad entre éstas, dificulta la automatización de estas actividades.

Capítulo 5

Fundamentos metodológicos y técnicos

En el capítulo anterior se identificaron algunos de los problemas existentes en la gestión de procesos software: la falta de alineamiento entre las definiciones de procesos y las herramientas de soporte, y la complejidad en la evaluación automatizada de procesos software. Con el objetivo de ofrecer respuesta a esta problemática, en la presente tesis se propone un marco de trabajo para el despliegue y evaluación de procesos software. Este marco se basa en la aplicación de unos paradigmas metodológicos y técnicos dentro de la Ingeniería del Software, como son la Ingeniería del Software dirigida por modelos o MDE y la integración de información mediante datos abiertos enlazados o LOD. Estos dos enfoques, a priori independientes, pueden complementarse, ya que una de las posibles aplicaciones de MDE es precisamente la integración de aplicaciones, así como ofrecer soporte para el diseño de ontologías [68].

En este capítulo se va a hacer un recorrido por los aspectos fundamentales de ambos paradigmas. En primer lugar, se describe la Ingeniería del Software dirigida por modelos, su aplicación para el desarrollo de software, la visión particular del consorcio OMG y las tecnologías que ofrece la comunidad Eclipse para este paradigma. Posteriormente, se describen los patrones comunes para la integración de información, el enfoque LOD, las ontologías o vocabularios relacionados con el proceso software y finalmente, algunas de las tecnologías existentes para el desarrollo de soluciones de integración.

5.1. Ingeniería del Software dirigida por modelos

Habitualmente los modelos en Ingeniería del Software son utilizados para especificar las características de los sistemas a desarrollar, comprender y analizar los requisitos del cliente, detectar errores con antelación y guiar las posteriores fases de desarrollo del software, en especial la programación.

Para el diseño de los modelos es importante disponer de un lenguaje de modelado que sea fácil de entender y sencillo de utilizar, con una sintaxis y una semántica bien definidas y que proporcionen un cierto nivel de abstracción mediante la ocultación de elementos no relevantes para el objetivo de modelado. Sin embargo, el modelado tradicional presenta algunos problemas importantes. Por un lado, la falta de mecanismos para validar u optimizar modelos y, por otro, la falta de sincronía entre los modelos y el código de las aplicaciones. Además, el modelado de sistemas es una actividad que puede llegar a consumir mucho tiempo, dependiendo de la complejidad del sistema bajo estudio. Todo esto resulta en que el retorno de la inversión que se obtiene al realizar modelos sea a menudo insuficiente.

Con el objetivo de superar estas barreras y dar una nueva oportunidad al papel del modelado durante el proceso software, aparece la Ingeniería del Software dirigida por modelos o MDE. Este paradigma promueve el uso de modelos como artefactos software, esto es, como elementos de primer nivel durante el ciclo de vida del software [177]. Al igual que otros enfoques recientes en Ingeniería del Software, MDE pretende mejorar la productividad, reduciendo los tiempos de desarrollo y el número de errores. Para ello, se define un conjunto de métodos, técnicas y herramientas destinadas a construir software de forma más rápida y sencilla, mediante el desarrollo y la transformación automática de modelos. Gracias a MDE, podemos tener editores, optimizadores, validadores y compiladores (mediante motores de transformación) de modelos.

Este enfoque de ingeniería puede ser utilizado en diferentes ámbitos [25]. Por ejemplo, para el modelado y gestión de procesos software, ya descrito en capítulos anteriores; para el diseño y generación automática de casos de prueba de software o *Model-Driven Testing* (MDT); y para la modernización de software, mediante técnicas de ingeniería inversa y directa de código, entre otras aplicaciones. Relativo a la modernización de software, destaca especialmente la propuesta formal de la OMG, denominada *Architecture-Driven Modernization* (ADM).

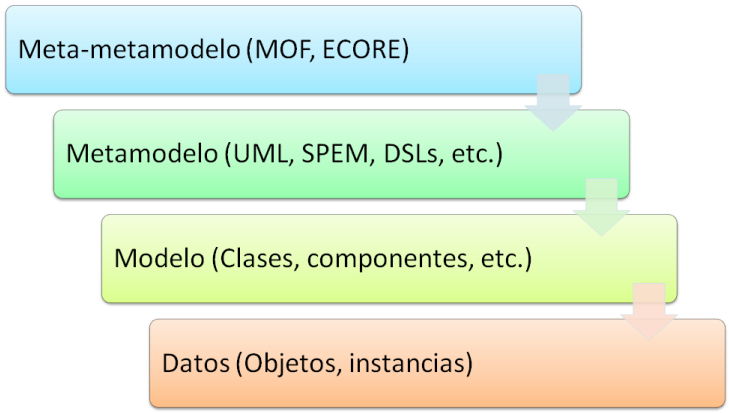

Figura 5.1: Arquitectura de cuatro niveles de modelado

5.1.1. Desarrollo de software dirigido por modelos

De todas las posibles aplicaciones de MDE, la ingeniería directa o generación automática de código desde modelos es, sin duda, la aplicación de MDE más extendida y desarrollada. El desarrollo dirigido por modelos o *Model-Driven Development* (MDD) tiene por objetivo la construcción de software mediante el desarrollo y la transformación sistemática de modelos, elevando así el nivel de abstracción requerido para el desarrollo de sistemas [10]. Así pues, podemos considerar a MDD como un subconjunto de MDE, en el sentido en que MDE va más allá de las actividades de desarrollo e incluye también otros procesos adicionales de la ingeniería del software.

Uno de los aspectos fundamentales en este paradigma es la arquitectura o pirámide de cuatro niveles de modelado, que podemos ver en la figura 5.1. Los niveles propuestos en la arquitectura son:

- *Datos (Nivel 0)*: Representa la información que maneja un sistema. Los datos son conformes a un determinado modelo.

- *Modelo (Nivel 1)*: Descripción o especificación abstracta de un sistema o parte de él y realizado con un lenguaje de modelado. Todo modelo es conforme a un metamodelo.

- *Metamodelo (Nivel 2)*: Definición de los constructores y reglas necesarias para definir la semántica de los modelos. Un metamodelo define un lenguaje de modelado general, como UML para representar sistemas software, o específico, *Domain Specific Language* (DSL), como BPEL para la composición de servicios web. Todo metamodelo es conforme a un meta-metamodelo.

- *Meta-metamodelo (Nivel 3)*: Definición de un lenguaje de modelado para el diseño de metamodelos. Un meta-metamodelo es conforme a sí mismo.

5.1.2. Model Driven Architecture

MDA es la visión particular del consorcio OMG para el desarrollo dirigido por modelos [95]. Esta metodología define un enfoque de desarrollo basado en diferentes niveles de abstracción y un conjunto de estándares para el modelado. La metodología tiene por objetivo construir sistemas software de forma sencilla y consiguiendo altos niveles de calidad, a través del diseño y refinamiento de modelos y la aplicación de transformaciones automáticas entre ellos. Los niveles considerados en MDA son:

- *Modelos independientes de computación o Computation Independent Model* (CIM), como por ejemplo un modelo de procesos de negocio en BPMN.

- *Modelos independientes de la plataforma o Platform Independent Model* (PIM), como por ejemplo un diagrama de casos de uso de un sistema informático.

- *Modelos específicos de la plataforma o Platform Specific Model* (PSM), como por ejemplo un diagrama de clases de diseño de un sistema web utilizando la plataforma *Java Enterprise Edition* (JEE).

- *Código*, como por ejemplo un conjunto de ficheros ejecutables en *C++*.

El proceso de desarrollo de MDA se basa en el refinamiento paso a paso de modelos, desde los más abstractos CIM o PIM hasta el código, pasando por los más específicos con detalles de implementación o PSM. La idea fundamental de esta metodología es que la transformación de los modelos se realice de forma automática, mediante la definición de un conjunto de reglas de transformación que definen cómo los elementos del modelo origen se transforman en elementos del modelo destino. Sin embargo, se requiere intervención humana a la hora de generar la definición inicial y después de cada transformación, para mejorar o

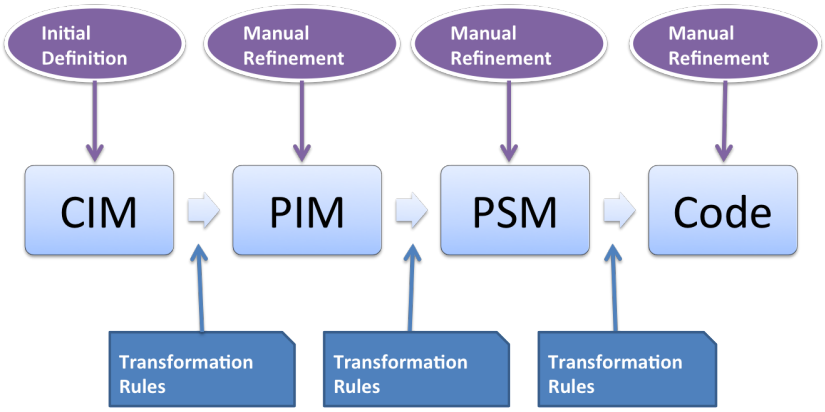

Figura 5.2: Proceso de desarrollo MDA

completar los modelos resultantes. Podemos observar una representación visual de este proceso en la figura 5.2.

Con respecto a los estándares englobados bajo MDA, podemos considerar: MOF, como infraestructura de meta-metamodelado necesaria para construir lenguajes de modelado; UML, como lenguaje principal para el desarrollo de los modelos en diferentes niveles; OCL, como lenguaje para la definición de restricciones y consultas sobre los modelos; *Query, Views and Transformations* (QVT), como lenguaje para definir reglas de transformación entre modelos (M2M); *MOF Model to Text Transformation Language* (MOFM2T), como lenguaje para definir transformaciones de modelo a texto; y *XML Metadata Interchange* (XMI), como formato de serialización de modelos. Además de los anteriores, la OMG desarrolló otros lenguajes de modelado en el contexto de MDA y complementarios a UML, como son el propio SPEM y los lenguajes *Common Warehouse Metamodel* (CWM) y *Ontology Definition Metamodel* (ODM).

5.1.3. Tecnologías Eclipse para el desarrollo dirigido por modelos

Además de los estándares de la OMG, existen multitud de tecnologías provenientes de la fundación *Eclipse*. *Eclipse Modeling Project*[1] tiene por misión

[1] http://www.eclipse.org/modeling

promocionar y evolucionar las tecnologías de desarrollo basadas en modelos, proporcionando herramientas, *frameworks* e implementaciones de los estándares [71]. A continuación, se relatan las principales áreas de este proyecto y algunas de las tecnologías relacionadas.

- *Desarrollo de sintaxis abstracta*: *Eclipse Modeling Framework* (EMF)[2] es el *framework* principal para el modelado y generación de código, orientado a la construcción de aplicaciones en *Java*. Incluye el lenguaje *Ecore* que implementa parcialmente el estándar de meta-metamodelo MOF.

- *Desarrollo de sintaxis concreta*: La herramienta *Graphical Modeling Framework* (GMF)[3] proporciona los componentes necesarios para desarrollar lenguajes gráficos y la herramienta *Xtext*[4] hace lo propio pero focalizado en la generación de lenguajes textuales.

- *Transformaciones de modelos*: *Atlas Transformation Language* (ATL)[5] es el lenguaje y el motor de ejecución de transformaciones entre modelos (M2M) más extendido, y el *framework Acceleo*[6] se especializa en la generación de código (M2T) desde modelos.

- *Modernización de software*: *MoDisco*[7] es el *framework* de *Eclipse* para el desarrollo de procesos de modernización de software para la mejora de sistemas, migración de componentes, extracción de documentación o análisis de métricas.

- *Herramientas de desarrollo de modelos*: Se incluyen implementaciones de los estándares de referencia UML, OCL y BPMN.

En el capítulo 6 se describe como el enfoque de desarrollo basado en niveles propuesto en la metodología MDA y las tecnologías de Eclipse para el desarrollo de software dirigido por modelos serán utilizadas para el despliegue de procesos software.

[2] http://www.eclipse.org/modeling/emf/
[3] http://www.eclipse.org/modeling/gmp/
[4] http://www.eclipse.org/Xtext/
[5] http://www.eclipse.org/atl/
[6] http://www.eclipse.org/acceleo/
[7] http://www.eclipse.org/MoDisco/

5.2. Integración de información

La integración de información es uno de los principales desafíos en Ingeniería del Software [74]. Esto se debe principalmente a: (i) la ausencia o disparidad de mecanismos para ofrecer la información en un formato común procesable por las máquinas y (ii) las discrepancias entre los modelos de datos utilizados por los sistemas a integrar. Habitualmente, las herramientas o sistemas informáticos están orientados hacia el usuario final para el desempeño de sus actividades en el contexto de una organización o de su vida diaria. Sin embargo, no todas estas aplicaciones están concebidas para una explotación automatizada de los datos por parte de otras aplicaciones y sin intervención del usuario.

Desde el punto de vista de la integración de información, existen dos estrategias o patrones principales para su implementación: *Enterprise Information Integration* (EII) y *Extract, Transform and Load* (ETL). La primera estrategia se basa en una arquitectura compuesta por un esquema global y una serie de fuentes externas de datos [108]. Las fuentes externas almacenan los datos reales, mientras que el esquema global ofrece una vista virtual e integrada de la información diseminada en cada una de las fuentes subyacentes. La segunda estrategia no utiliza una vista virtual, sino un repositorio central o *data warehouse* que almacena los datos previamente extraídos de las fuentes externas y transformados en un modelo común [93].

Con independencia de la estrategia de integración utilizada, toda solución de integración requiere de: (i) envoltorios o *wrappers* que permitan extraer o publicar los datos de los sistemas; (ii) procesos de integración para transformar y transportar los mensajes de datos; (iii) contenedores de información basados en vistas lógicas, en el caso de EII, o en almacenes de datos, en el caso de ETL y (iv) interfaces para la consulta de los datos.

5.2.1. Datos abiertos enlazados

La Web es un sistema de distribución de documentos de hipertexto accesible a través de Internet, que permite compartir e intercambiar información entre diferentes sistemas. Sin embargo, la Web está dirigida principalmente para consumo humano.

Iniciativas emergentes como la Web Semántica o LOD se están utilizando para la integración de información en la Web y la construcción de terceras aplicaciones que permitan obtener un mayor valor añadido. Estas iniciativas se centran en la publicación, intercambio, recolección y consulta de los contenidos existentes en los sistemas web, de modo que puedan ser fácilmente procesados

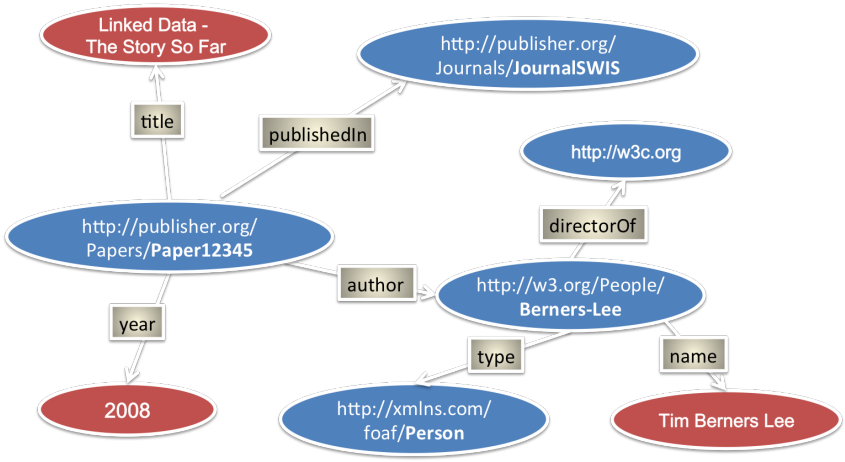

Figura 5.3: Modelo de ejemplo en RDF

mediante programas software [20].

Con el objetivo de publicar de manera estructurada y uniforme los contenidos de la web, se propuso *Resource Description Framework* (RDF). RDF es una especificación de la *World Wide Web Consortium* (W3C) que ofrece un método para describir unívocamente recursos de información existentes en la web [96]. El modelo de datos de RDF se basa en el establecimiento de sentencias sobre recursos. Estas sentencias, en forma de tripletas sujeto-predicado-objeto, permiten describir cualquier aspecto o característica de un determinado recurso en la Web, así como describir relaciones entre éstos. De este modo, un conjunto de sentencias RDF representa un grafo dirigido.

En la figura 5.3 podemos observar un modelo de ejemplo RDF con información relativa a Tim Berners Lee, precursor del enfoque LOD. En este modelo se aprecia un recurso referido a él, de tipo *Person* y director de una organización, en este caso, la propia W3C. Asimismo, podemos observar cómo el artículo de título *Linked Data - The Story So Far* fue publicado en una revista por él en el año 2008.

Gracias a este sencillo modelo conceptual en forma de grafo, los sistemas que quieren formar parte de una solución de integración basada en LOD deben exponer sus datos siguiendo alguno de los formatos de serialización soportados

para RDF, como son XML, *N3*, *Turtle*, *N-Triples* o *JSON-LD*. Utilizando un formato estandarizado y común como RDF, se evita el problema de la heterogeneidad en los formatos utilizados por los distintos sistemas. Con respecto al problema de las diferencias en los modelos de datos utilizados por las distintas fuentes de información, se propone la utilización de ontologías o vocabularios compartidos, que permiten describir la estructura de la información de algún determinado ámbito o dominio de conocimiento [73].

5.2.2. Ontologías y vocabularios

Una ontología es una especificación explicita y formal de una conceptualización compartida [72]. Las ontologías permiten representar el conocimiento relativo a un determinado dominio, recopilando los conceptos de interés, sus propiedades y sus relaciones atribuibles. Las ontologías definen un vocabulario compartido tanto para los humanos, como para los sistemas informáticos implicados en ciertas áreas de conocimiento.

Podemos distinguir entre ontologías de dominio y ontologías superiores. Las primeras representan exclusivamente conceptos aplicables a una determinada parte del conocimiento universal, mientras que las segundas están destinadas a representar conceptos aceptados en un amplio rango de dominios. Podemos citar a las ontologías *Gene Ontology* (GO) [9] y *Suggested Upper Merged Ontology* (SUMO) [132], como ejemplos de una ontología para el dominio de la genética y una ontología para la representación de conceptos generales.

A continuación, se describen los principales vocabularios relacionados con el software encontrados hasta la fecha:

- *Description Of A Project* (DOAP)[8]. Vocabulario para describir proyectos software, en especial para software de fuentes abiertas.
- *Asset Description Metadata Schema for Software* (ADMS.SW)[9]. Vocabulario creado con propósitos similares al anterior.
- *SourceForge Trove Map*[10]. Vocabulario utilizado por la plataforma *SourceForge* para categorizar proyectos con respecto a su licencia, lenguaje de programación y otros aspectos.

[8] https://github.com/edumbill/doap/wiki
[9] https://joinup.ec.europa.eu/asset/adms_foss/
[10] http://sourceforge.net/apps/trac/sourceforge/wiki/Software%20Map%20and%20Trove

- *Software Package Data Exchange* (SPDX)[11]. Vocabulario dirigido a estandarizar la forma en la cual las organizaciones publican los metadatos relativos a las licencias de distribución y uso del software.

- *ISO 19770-2*[12]. Especificación desarrollada por la ISO con el objetivo de optimizar la identificación y gestión del software a través de un etiquetado común.

- *Open Services for Lifecycle Collaboration* (OSLC)[13]. Conjunto de especificaciones dirigidas a mejorar la integración de plataformas ALM.

5.2.3. Tecnologías para datos abiertos enlazados

Alrededor del enfoque LOD se han ido desarrollando un conjunto de tecnologías y herramientas para la publicación y consumo de datos RDF. A continuación, se presentan algunas de ellas:

- *Diseño de ontologías*: Existen diferentes lenguajes para el diseño de ontologías, desde los más tradicionales como *F-logic* [92], hasta los desarrollados específicamente para la Web en el contexto de la W3C, como *RDF Schema*, o el más expresivo *Web Ontology Language* (OWL) [5]. Para un diseño efectivo se requiere de herramientas dedicadas, como *Protégé*[14] o *Neologism*[15], entre otras.

- *Almacenamiento de datos*: De cara el soporte persistente de la información RDF se necesitan repositorios dedicados, como *OpenRDF Sesame*[16] o *Virtuoso*[17].

- *Razonamiento*: *Pellet*[18] o *KiWi Reasoner*[19] son algunos ejemplos de componentes software que permiten inferir consecuencias lógicas a partir de un conjunto de tripletas RDF.

[11] http://spdx.org/
[12] http://www.iso.org/iso/catalogue_detail.htm?csnumber=53670
[13] http://open-services.net/specifications/
[14] http://protege.stanford.edu/
[15] http://neologism.deri.ie/
[16] http://www.openrdf.org/
[17] http://virtuoso.openlinksw.com/
[18] http://clarkparsia.com/pellet/
[19] http://marmotta.incubator.apache.org/kiwi/reasoner.html

- *Exposición de datos*: Existen dos mecanismos para exponer información siguiendo el enfoque LOD. Por un lado, haciendo uso del estilo arquitectónico *Representational State Transfer* (REST) [59] para acceder, actualizar, crear o borrar recursos concretos mediante el protocolo *Hypertext Transfer Protocol* (HTTP)[20], o bien, utilizando *SPARQL Protocol and RDF Query Language* (SPARQL)[21] para consultar y manipular grafos RDF.

- *Consumo de datos*: La mayor parte de los lenguajes de programación actuales disponen de librerías para hacer un consumo eficaz de RDF. Podemos citar a *Apache Jena*[22] para *Java* o *RDFlib*[23] para *Python*, entre otras.

Además de las herramientas anteriores, existen algunas plataformas como *Linked Media Framework*[24], que integran diversos servicios como almacenamiento, razonamiento, anotación automática, motor de consultas y búsquedas semánticas, caché, versionado, etc.

La integración de información haciendo uso del enfoque y de las tecnologías LOD nos permitirá abordar el problema de la evaluación de procesos software, mediante la apertura de datos de sistemas de apoyo al proceso software, como veremos en el capítulo 6.

[20] http://www.w3.org/TR/ldp/
[21] http://www.w3.org/TR/sparql11-overview/
[22] http://jena.apache.org/
[23] http://www.rdflib.net/
[24] https://code.google.com/p/lmf/

Capítulo 6

Marco de trabajo

En este capítulo se presenta *Software Process Deployment & Evaluation Framework* (SPDEF), un marco de trabajo para el despliegue y evaluación de procesos software sobre herramientas de soporte. Este *framework* persigue facilitar la adaptación de los entornos software de soporte a la producción y gestión de software, mejorando así la consistencia entre las definiciones de los procesos y su instanciación real. Asimismo, el framework pretende simplificar la construcción de mecanismos que permitan evaluar los procesos software de forma automática.

Para el despliegue de los procesos, se propone el uso del enfoque MDD para automatizar la adaptación de las herramientas de soporte a partir de las definiciones de los procesos software. Con respecto a la evaluación de los procesos, se propone la utilización del enfoque LOD para el desarrollo de soluciones de integración de los datos expuestos por las herramientas de soporte. A continuación, se describen los elementos que forman este *framework*: un método sistemático para el despliegue y evaluación de los procesos, un conjunto de modelos, las relaciones entre éstos y una serie de herramientas de apoyo.

6.1. Método para el despliegue y evaluación

Este método se compone de cuatro actividades, en las que participan dos roles diferenciados y haciendo uso de cinco tipos de herramientas. Las actividades previstas en este método, que aparecen en la figura 6.1, son las siguientes:

- *Modelado de procesos software*, para la definición de las metodologías y procesos de desarrollo o mantenimiento de software.

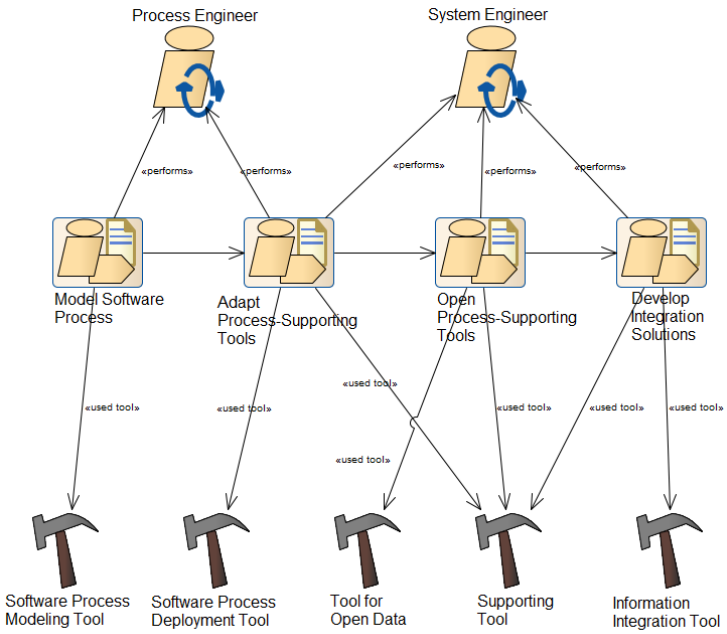

Figura 6.1: Método general para el despliegue y evaluación de procesos software

- *Adaptación de herramientas de soporte*, para alinear las definiciones de procesos con la utilización efectiva en las herramientas.
- *Apertura de herramientas de soporte*, para conseguir que las herramientas expongan de forma homogénea la información generada durante la ejecución de los proyectos software.
- *Desarrollo de soluciones de integración*, para desarrollar aplicaciones que permitan evaluar aspectos del proceso software, mediante la integración de los datos publicados por las herramientas de soporte.

Para la realización de las actividades se ha contemplado la participación de los siguientes perfiles:

- *Ingeniero de Procesos*, responsable de definir y modelar los procesos y

metodologías necesarias para producir software en una determinada organización.

- *Ingeniero de Sistemas*, responsable de instalar y preparar las herramientas de soporte a la gestión o producción de software.

Para poder llevar a cabo este método de forma eficaz, se necesita una serie de herramientas auxiliares:

- *Herramienta de modelado de procesos software*, para poder diseñar los procesos y metodologías de desarrollo o mantenimiento de software. Cualquier herramienta compatible con SPEM sería válida.
- *Herramienta de despliegue de procesos software*, para poder adaptar automáticamente las herramientas de soporte al proceso software. Este *framework* incluye una herramienta diseñada específicamente para estas labores.
- *Herramienta de soporte*, para asistir a los miembros de los equipos de proyecto en el desempeño de las actividades durante el ciclo de vida del software, como por ejemplo, las herramientas de edición de código, los editores de modelos UML, etc.
- *Herramienta para la apertura de datos*, para exponer la información gestionada por las herramientas de soporte en algún formato de datos estructurado.
- *Herramienta para la integración de información*, para la construcción de los componentes software necesarios para consumir los datos ofrecidos por las herramientas para la apertura de datos. En este punto, se podrían considerar aquellas herramientas que permitan implementar procesos de integración de datos o ejecutar consultas distribuidas sobre diferentes conjuntos de datos.

A continuación, se describen en detalle cada una de las actividades que forman parte del método propuesto.

6.1.1. Modelado de procesos software

SPEM es el lenguaje más extendido para la definición de procesos software. A pesar de ciertas carencias evidenciadas en el lenguaje, el estándar sigue siendo un lenguaje adecuado para la representación de los procesos software. Por ello,

SPEM será el lenguaje seleccionado para la representación de procesos software en este *framework*, además de por ser el más extendido, por utilizar la arquitectura estándar de meta-metamodelado de cuatro niveles.

El Ingeniero de Procesos de la organización será el responsable de definir los procesos a seguir durante la ejecución de los proyectos de desarrollo o mantenimiento de software. Para llevar a cabo esta actividad, se precisa el uso de una herramienta de modelado de procesos software compatible con SPEM.

6.1.2. Adaptación de herramientas de soporte

Esta actividad tiene por objetivo adaptar semi-automáticamente las herramientas de soporte al proceso software, a partir de lo explicitado en el modelo de proceso definido con SPEM. Para ello se emplearán los principios y las técnicas del paradigma MDE y se seguirá un proceso similar al planteado en la metodología MDA.

En la figura 6.2 se puede observar el conjunto de actividades a realizar por parte del Ingeniero de Procesos y del Ingeniero de Sistemas. Son las siguientes:

- *Derivar modelos de despliegue del proceso software:* En esta actividad, el Ingeniero de Procesos hará uso de las posibilidades que ofrecen los motores de transformación M2M para obtener automáticamente modelos de despliegue de procesos software a partir de los modelos de proceso en SPEM.
- *Refinar modelos de despliegue del proceso software:* El Ingeniero de Procesos podrá extender y manipular los modelos de despliegue generados automáticamente, para mejorarlos y adecuarlos a las particularidades de la metodología y la organización.
- *Derivar modelos de herramientas genéricas:* Estos modelos podrán generarse de forma automática a partir de transformaciones M2M sobre los modelos de despliegue. Esta actividad, dirigida al Ingeniero de Sistemas, tiene por objetivo obtener los modelos que representen los aspectos principales de los tipos de herramientas a utilizar en los proyectos.
- *Componer modelos de herramientas específicas:* Esta actividad tiene por objetivo detallar los aspectos tecnológicos de cada una de las herramientas a utilizar para dar soporte eficaz al proceso software. Dado que algunas herramientas específicas pueden incluir características de varios tipos de herramientas, el Ingeniero de Sistemas tendrá que construir los modelos específicos mediante la composición de los modelos generados con la actividad anterior.

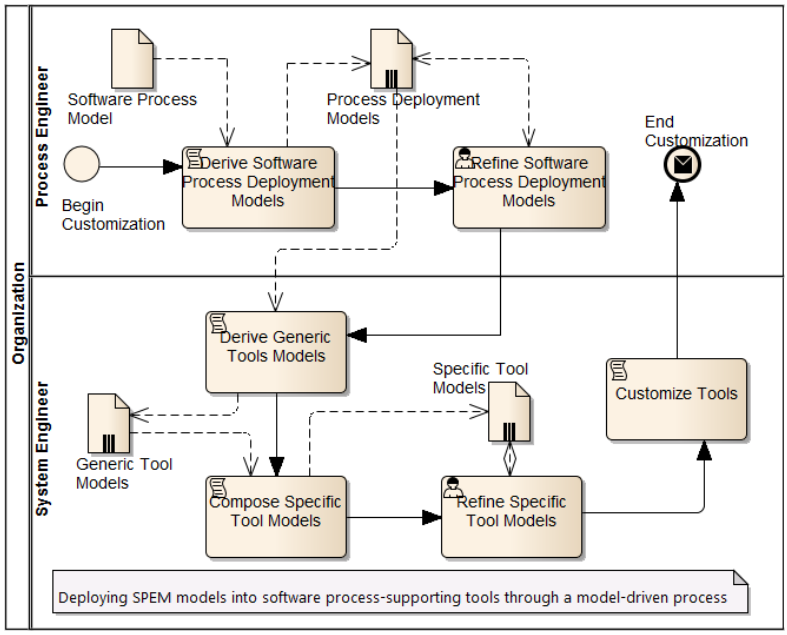

Figura 6.2: Adaptación de herramientas de soporte

- *Refinar modelos de herramientas específicas:* Los modelos de herramientas específicas, compuestos a partir de modelos genéricos, podrán ser mejorados y especializados para adecuarlos a su despliegue final sobre las herramientas de soporte.

- *Configurar las herramientas de soporte:* Como último paso de este proceso, el Ingeniero de Sistemas debe configurar cada una de las herramientas de soporte. Para ello, hará uso de un motor de transformación M2T para lanzar los scripts necesarios para adaptar las herramientas seleccionadas.

El conjunto de actividades relativas a la adaptación de las herramientas de soporte tienen por objetivo, en última instancia, configurar automáticamente las herramientas necesarias en el arranque de cada proyecto, partiendo en origen de una descripción SPEM en alto nivel del proceso software. De esta manera, siguiendo un proceso de transformación automático y de refinamiento manual

de modelos, se podrán obtener las tareas, esqueletos de artefactos software y plantillas necesarias para gestionar eficazmente el ciclo de vida del software mediante las herramientas de soporte. El objetivo de esta actividad es, por tanto, la parametrización de ciertos tipos de herramientas de soporte, a diferencia de MDA, cuyo fin es la construcción de sistemas software.

La adaptación de las herramientas de soporte se basa en las oportunidades que ofrece la transformación automática de modelos, inspirándose en ciertas características de algunas de las herramientas del mercado como *IRIS Process Author* o *EPF Composer*, así como de algunos de los trabajos recopilados en el capítulo 3, como [30] o [117]. Sin embargo, a diferencia de las herramientas y los trabajos indicados anteriormente, el método de adaptación que se propone es independiente de las plataformas software utilizadas y agnóstico del tipo de proceso o metodología de desarrollo.

Una aplicación parcial de este método pero orientado al ámbito de los procesos y entornos de aprendizaje, se encuentra en [48]. En posteriores secciones se presentan el detalle de los modelos implicados, las relaciones entre éstos y los componentes software de apoyo al despliegue.

6.1.3. Apertura de herramientas de soporte

Las diferentes herramientas de soporte gestionan la información derivada de la ejecución de las actividades del ciclo de vida del software. Si las diferentes herramientas de soporte, ya sean primarias o secundarias, publicasen sus datos de forma normalizada y estandarizada, se podría simplificar la construcción de nuevas herramientas focalizadas en la evaluación de la calidad de los procesos. Así pues, el objetivo es conseguir que las herramientas publiquen sus datos en un formato común y fácilmente procesable por las máquinas, de tal modo que no sean silos de información.

Uno de los problemas importantes a la hora de integrar la información generada en los proyectos es la discrepancia con respecto a los modelos de datos utilizados en las diferentes herramientas. Es fundamental conseguir que la semántica acompañe a los propios datos publicados por las herramientas, para así facilitar los posteriores procesos de alineamiento o *mapping* de la información expuesta en las diferentes herramientas. El análisis de datos procedentes de proyectos desplegados en sistemas de soporte al proceso software es todavía un área incipiente en Ingeniería del Software.

En esta investigación se propone la utilización del enfoque LOD como medio para la integración de los datos gestionados por las distintas herramientas [158, 161]. En este contexto, el Ingeniero de Sistemas implementará o utilizará los

componentes software indicados en este *framework* para la apertura de los datos gestionados en las herramientas.

A la hora de exponer los datos desde las herramientas de soporte, hay que tener en cuenta si la herramienta es de código abierto o privativo, si ofrece sus datos mediante algún mecanismo procesable por las máquinas o no, y si se utilizan formatos propietarios o libres. Hoy día existe mucha variabilidad con respecto a estos factores. En el mejor de los casos, si la herramienta de soporte ofrece una *Application Programming Interface* (API) basada en LOD, entonces estaría lista para ser utilizada en el contexto de la evaluación de procesos. En el extremo opuesto, podemos encontrarnos con herramientas totalmente cerradas en cuanto a los datos que almacenan.

A continuación, se detallan los mecanismos a implementar en función de una serie de decisiones implicadas para conseguir que las herramientas se conviertan en conjuntos de datos o *datasets* RDF. Lo podemos observar en la figura 6.3.

- *Implementar un adaptador de API*: En ocasiones las aplicaciones web de soporte al proceso software incorporan algún tipo de API para permitir la reutilización de sus datos, habitualmente siguiendo el enfoque REST. En estos casos habría que desarrollar un adaptador que reciba peticiones LOD y haga las invocaciones adecuadas a dichas APIs, realizando las conversiones de datos necesarias para ello.

- *Implementar una extensión del software*: En aquellos casos donde la herramienta a integrar en LOD no disponga de una API, pero sí se dispone de acceso para modificar el código fuente de la misma, entonces sería necesario implementar una extensión a la aplicación. Construir una extensión para un software en particular con el objetivo de publicar LOD no siempre es una tarea sencilla y además suele ser una labor costosa de implementar. Para dar respuesta a esta problemática, se propone generar automáticamente los metadatos RDF y exponerlos de forma controlada, a partir del modelo de las aplicaciones web. Esta aproximación, descrita en [49], se basa en la introspección automática del código fuente de aplicaciones basadas en el patrón arquitectónico *Model View Controller* (MVC) para descubrir las entidades y atributos que forman parte del modelo. A partir de dicha introspección, se podrán generar datos RDF conformes a un vocabulario local generado automáticamente desde dicho modelo. Asimismo, para consumir esos datos, se debe proporcionar una interfaz de servicios enriquecida con algún mecanismo de autorización para controlar el nivel de acceso a los datos.

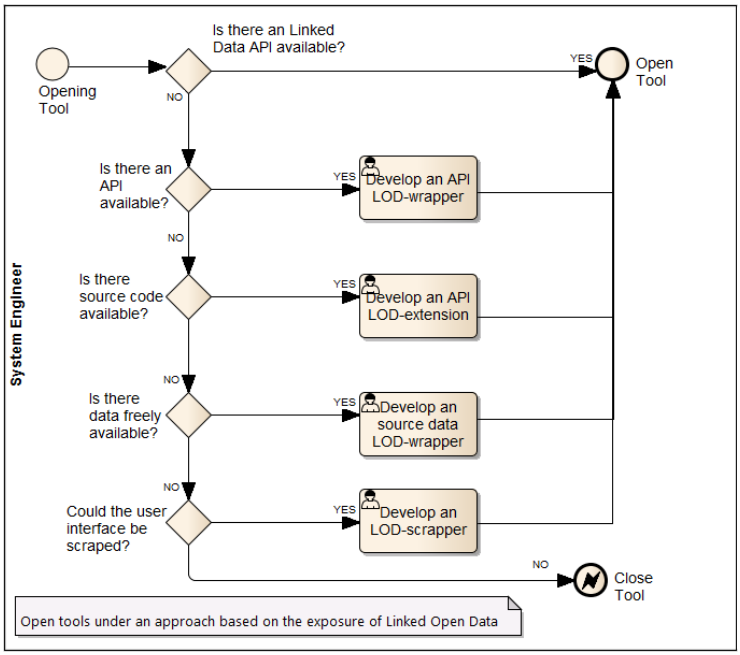

Figura 6.3: Apertura de herramientas de soporte

- *Implementar un adaptador de datos*: En otras ocasiones no se tiene acceso al código de la herramienta a integrar, aunque sí se tiene acceso a los datos. En estos casos se necesitaría desarrollar un adaptador que, recibiendo peticiones LOD, las traduzca en las consultas pertinentes sobre la base de datos, ficheros estructurados, *logs* o cualquier otro mecanismo de persistencia de información que utilice la aplicación.

- *Implementar un extractor de datos*: Los sistemas cerrados tanto a nivel de código como a nivel de datos suelen ser más férreos a la hora de integrarlos como parte de una solución LOD, ya que en su construcción no fueron diseñados para su reutilización por parte de terceras aplicaciones. En estos casos, hay que utilizar técnicas complejas de extracción o *scraping* para extraer los datos [147] y/o usar anotadores semánticos [86].

Una aplicación parcial de este método pero dirigido al ámbito de los sistemas de información científica, se encuentra en [85]. En la literatura podemos encontrar algunos trabajos relacionados, pero con objetivos diferentes a los planteados en esta investigación, como en [32] donde se presenta un enfoque para la obtención y análisis de métricas recopiladas de diferentes fuentes de datos. Recientemente, diversas empresas proveedoras de software lideradas por IBM están desarrollando un conjunto de especificaciones [194] dirigidas a facilitar la integración de herramientas de desarrollo u operación de software haciendo uso de tecnologías LOD y servicios web REST.

6.1.4. Desarrollo de soluciones de integración

El Ingeniero de Sistemas será el responsable de la implantación de soluciones de integración de datos con el fin de evaluar los procesos software a partir de la información publicada por las herramientas de soporte.

El enfoque LOD permite construir tanto soluciones de integración ETL, mediante el consumo y almacenamiento de datos RDF, como soluciones EII, mediante consultas SPARQL federadas sobre diferentes *endpoints*. El consumo de los datos derivados de la ejecución de los procesos software nos ofrece una serie de posibilidades relativas a la evaluación que a continuación se describen.

Análisis de indicadores y métricas de software

La medición de software y el análisis de indicadores es una de las actividades que puede ser potenciada a través del enfoque de integración de datos. Por un lado, obteniendo nuevas métricas mediante la ejecución de consultas sobre los *datasets*, independientemente de si esas métricas están implementadas en la lógica de la aplicación y accesibles desde su interfaz de usuario y, por otro, calculando nuevas métricas derivadas a partir de mediciones recopiladas de herramientas distintas.

Un ejemplo de métrica indirecta, obtenida con este enfoque de integración, es el número de cambios registrados o *commits* en el sistema de control de versiones con respecto a los hitos o *milestones* definidos en la herramienta de monitorización de proyectos. De igual forma, resulta interesante conocer otras como por ejemplo, el número de compilaciones o *builds* de código realizadas en el sistema de integración continua por cada *milestone* o bien, el número de pruebas o *tests* definidos mediante la herramienta de gestión de pruebas por cada tarea o *issue* de mejora de software, etc.

Revisiones de calidad

La automatización de revisiones de calidad en proyectos de software es otro de los beneficios que aporta el enfoque LOD sobre las herramientas de soporte al proceso software. Para su automatización se requiere que los criterios de evaluación se definan en términos de consultas sobre los datos expuestos por las herramientas de soporte. Por ejemplo, comprobar que se ha elaborado el modelo conceptual de análisis de un determinado proyecto de desarrollo software puede traducirse en una consulta sobre el número de clases UML registradas en el *dataset* de la herramienta de modelado. De esta forma es posible comprobar automáticamente la adherencia de los proyectos a las descripciones de los procesos, estándares y procedimientos implantados en la organización.

6.2. Modelos

En este *framework* se utiliza un conjunto de modelos a diferentes niveles de abstracción: en el nivel CIM se emplea el estándar SPEM para la definición de procesos software; para el nivel PIM se han definido dos nuevos modelos orientados al despliegue de los procesos; y en el nivel PSM se definen modelos de herramientas genéricas y modelos de herramientas específicas. Los detalles sobre el lenguaje SPEM, la utilidad y el ámbito de cada paquete del metamodelo fueron descritos en la sección 2.2.

En función del espacio tecnológico donde estemos situados, los modelos se materializarán en artefactos diferentes, aunque manteniendo la misma semántica. Desde la perspectiva del despliegue de procesos a través del enfoque MDE, los modelos se implementan como metamodelos *Ecore*, mientras que desde el punto de vista de la evaluación de los procesos mediante el enfoque LOD, los modelos se implementan como vocabularios RDF. Hay que tener en cuenta que no se han implementado vocabularios para los modelos de herramientas específicas, ya que uno de los pilares fundamentales en el enfoque LOD es la reutilización de vocabularios compartidos y no específicos de cada sistema de información. En la sección 6.4 se ofrecen más detalles sobre la implementación de los modelos.

A continuación se describen los modelos específicamente diseñados para este marco de trabajo. Nótese que los modelos se presentan utilizando la notación de diagramas de clases de UML.

6.2.1. Modelos de despliegue de procesos

Como se demostró en el estudio de la literatura sobre los usos y aplicaciones del lenguaje SPEM [160], existe una importante brecha entre las posibilidades de modelado que ofrece este lenguaje y cómo los aspectos metodológicos de la gestión o producción del software pueden ser implantados en las herramientas de soporte. Con el objetivo de asimilar los elementos implicados en los modelos de procesos software con respecto a los conceptos y relaciones subyacentes en las herramientas de soporte, se han diseñado una serie de modelos de nivel PIM, que se describen a continuación.

Modelo de productos de trabajo software

Tradicionalmente durante el modelado de los procesos software, los productos de trabajo son tratados como unidades atómicas, con nombre pero sin estructura, que se generan o modifican durante el transcurso de los proyectos. De hecho, en el lenguaje SPEM no se dispone de mecanismos para detallar la estructura de los productos de trabajo, permitiendo sólo distinguir si se trata de un artefacto (*Artifact*), un entregable (*Deliverable*) o un resultado (*Outcome*) del proyecto.

Los productos de trabajo típicos en los procesos software suelen ser principalmente documentos o código fuente y suelen gestionarse desde herramientas especializadas o genéricas. Por ejemplo, las herramientas de modelado software, como *Rational Rose*[1], o los sistemas de control de versiones, como *Git*[2], son ejemplos de herramientas especializadas que ofrecen soporte a la gestión de ciertos tipos de productos de trabajo. Sin embargo, otras herramientas genéricas como los sistemas de gestión de contenidos, los sistemas de gestión documental o los sistemas de edición colaborativa o wiki también pueden utilizarse para albergar evidencias de los procesos, como en [43], donde utilizan *MediaWiki* para la especificación de requisitos software.

El diseño de modelos para ciertos tipos de productos de trabajo [55] ha sido una aproximación utilizada en diversas metodologías web dirigidas por modelos. Sin embargo, es habitual la necesidad de adaptar y personalizar metodologías conocidas para su aplicación en organizaciones específicas y para situaciones concretas, lo que se conoce como *process tailoring*. Por ello, se propone un modelo que permita definir productos de trabajo flexibles en cuanto a la estructura y tipo de sus artefactos. Lo podemos ver en la figura 6.4. Este modelo, denominado *Software Work Product Model* (SWPM), comprende los siguientes elementos:

[1] http://www-03.ibm.com/software/products/us/en/ratirosefami
[2] http://git-scm.com/

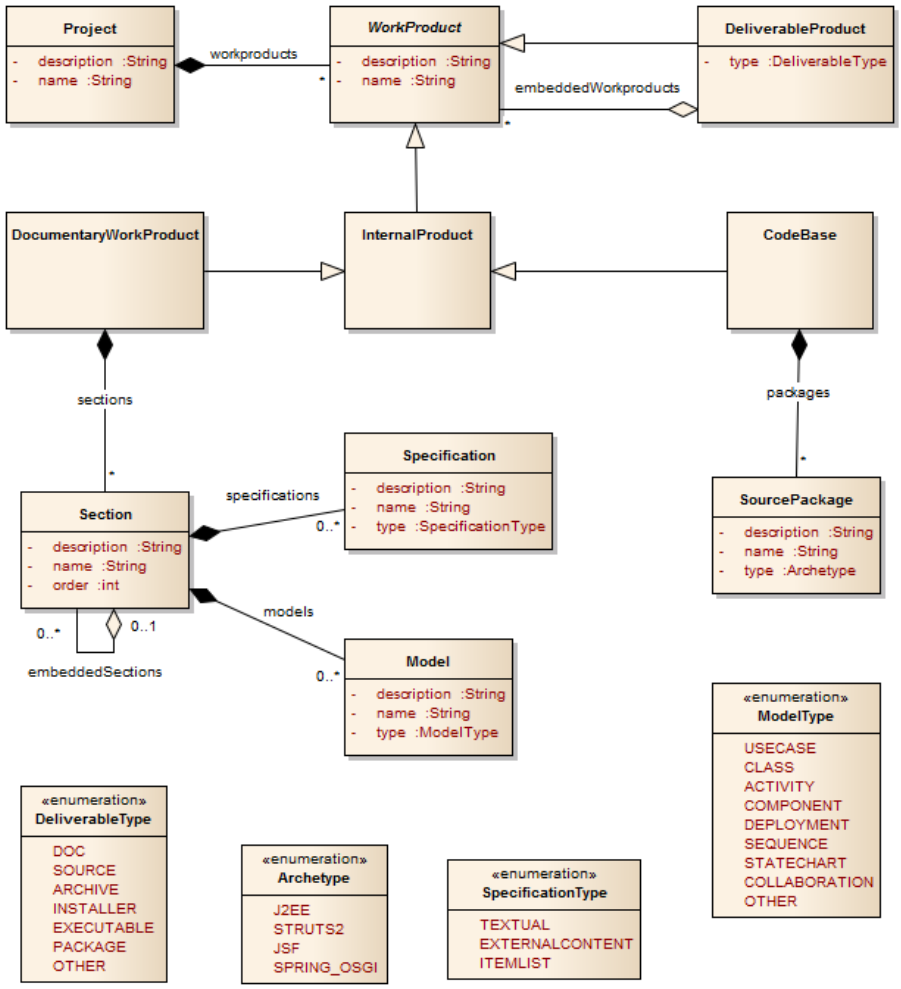

Figura 6.4: Modelo de productos de trabajo software

- *Project*. Clase contenedora de todos los productos de trabajo de un proyecto.
- *WorkProduct*. Clase abstracta similar al concepto de *WorkProduct* en SPEM que representa un determinado producto desarrollado o gestionado durante el proceso software. Se clasifican en productos entregables o internos.
- *DeliverableProduct*. Clase abstracta que representa un producto de trabajo desarrollado durante el transcurso de un proyecto software y con valor para terceros.
- *InternalProduct*. Clase abstracta que representa un producto interno al desarrollo o mantenimiento de software. Puede ser de tipo documental o de código.
- *DocumentaryWorkProduct*. Clase que representa un documento técnico de trabajo.
- *Section*. Clase que representa una sección dentro de un determinado documento generado durante el proyecto.
- *Specification*. Clase que representa una especificación, desarrollada sin utilizar un lenguaje de modelado, de algún aspecto del proyecto software.
- *Model*. Clase que representa un modelo diseñado haciendo uso un lenguaje de modelado.
- *CodeWorkProduct*. Clase que representa la base de código de un proyecto software.
- *SourcePackage*. Clase que representa un determinado paquete de código fuente basado en algún lenguaje de programación.
- *Archetype*. Enumeración con los posibles tipos de paquetes de código fuente. Engloba a aquellas plantillas predefinidas de código, siguiendo los arquetipos disponibles en sistemas de gestión de la construcción como *Maven* u otros.
- *ModelType*. Enumeración con los posibles tipos de modelos. Comprende los tipos propuestos en la especificación de UML, aunque podría englobar otros tipos de modelos desarrollados con otros lenguajes, ya sean genéricos, como *IDEF* [125], o de propósitos específicos, como *FlexoLD* [50].

- *SpecificationType*. Enumeración con los posibles tipos de especificaciones no basadas en modelos. Pueden consistir en una descripción textual, una lista de elementos o contenido binario gestionado por alguna herramienta externa.

- *DeliverableType*. Enumeración con los posibles tipos de entregables de un proyecto. Pueden ser ficheros empaquetados, instaladores o documentos, entre otros tipos.

Modelo de control de proyectos software

La planificación y monitorización de la ejecución de proyectos software puede llevarse a cabo con diferentes herramientas, desde sistemas basados en gestión de tareas, hasta herramientas especializadas en diagramas de Gantt o incluso mediante simples hojas de cálculo.

En la figura 6.5 se muestra el modelo diseñado teniendo en cuenta los elementos típicos a la hora de controlar los proyectos de desarrollo o mantenimiento de software. El modelo, *Software Project Control Model* (SPCM), comprende los siguientes elementos:

- *Project*. Clase contenedora de todos las tareas y milestones establecidos para la planificación y monitorización de un proyecto software.

- *Milestone*. Clase que representa los hitos del desarrollo o versiones del software.

- *Task*. Clase que define la acción de realizar una labor necesaria para conseguir los objetivos del proyecto.

- *TaskCategory*. Clase que permite categorizar una tarea según su ámbito de aplicación.

- *Role*. Clase que representa el papel que juega un determinado miembro de un proyecto de desarrollo o mantenimiento de software.

- *TaskDependency*. Clase de asociación que permite representar la dependencia entre dos tareas concretas.

- *DependencyType*. Enumeración con los diferentes tipos de dependencia entre tareas.

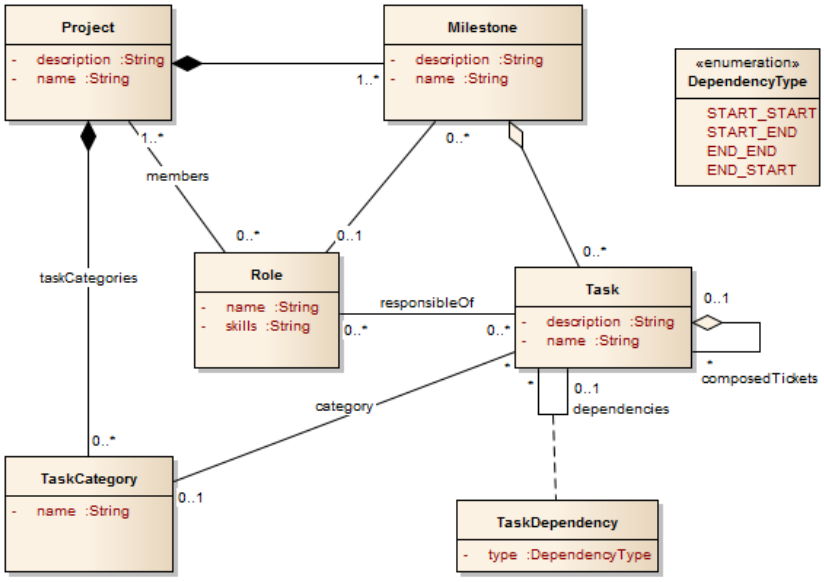

Figura 6.5: Modelo de control de proyectos software

6.2.2. Modelos de herramientas genéricas

Son varias las familias de herramientas que pueden proporcionar soporte a la ejecución de las actividades del ciclo de vida de los proyectos software. Estos modelos PSM están estrechamente vinculados con la estructura de la información gestionada por las herramientas de soporte. Para ilustrar los modelos de herramientas genéricas, a continuación se presentan tres modelos correspondientes a sistemas wiki, sistemas de edición de modelos visuales y sistemas de gestión de tareas.

Modelo de herramientas wiki

A partir del análisis de diversos sistemas, como *MediaWiki*, *Confluence*[3] o *DokuWiki*[4], se ha diseñado el modelo *WIKI tool Model* (WIKIM) de la figura

[3] https://www.atlassian.com/en/software/confluence
[4] https://www.dokuwiki.org/dokuwiki

6.6. Téngase en cuenta que este modelo no describe completamente el modelo conceptual de todos los tipos de sistemas wiki, sino sólo de aquellos elementos que se consideran de interés para la presente investigación. A continuación, se describen los elementos principales:

- *WikiDatabase*. Clase contenedora de todos los elementos de la wiki.
- *WikiContent*. Clase abstracta que representa a cualquier tipo de contenido que puede almacenar una wiki.
- *Category*. Clase que representa a una categoría dentro una instancia de la wiki, con la cual se pueden clasificar artículos.
- *User*. Clase que representa al usuario y a la página del mismo dentro de la wiki.
- *File*. Clase que representa a un determinado archivo binario almacenado en la wiki.
- *Article*. Clase que representa a un artículo dentro de la wiki.
- *Section*. Clase que permite declarar una sección dentro de un artículo de la wiki.
- *SectionContent*. Clase abstracta que se especializa en cada tipo de contenido que puede incluirse dentro de una sección de un artículo.
- *Image*. Clase que representa la inclusión de una imagen dentro de un artículo wiki.
- *Paragraph*. Clase que representa un párrafo de texto.
- *ItemList*. Clase que representa un lista de items.
- *Item*. Clase que representa el item de una lista.

En la literatura podemos encontrar algunos trabajos relacionados que describen la utilización de las wikis como medio para representar productos de trabajo comunes en el proceso software [3, 43, 115]. Otro trabajo relacionado se presenta en [46], donde los autores proponen un DSL para la generación de instancias de wikis contextualizadas para las estrategias corporativas de las organizaciones.

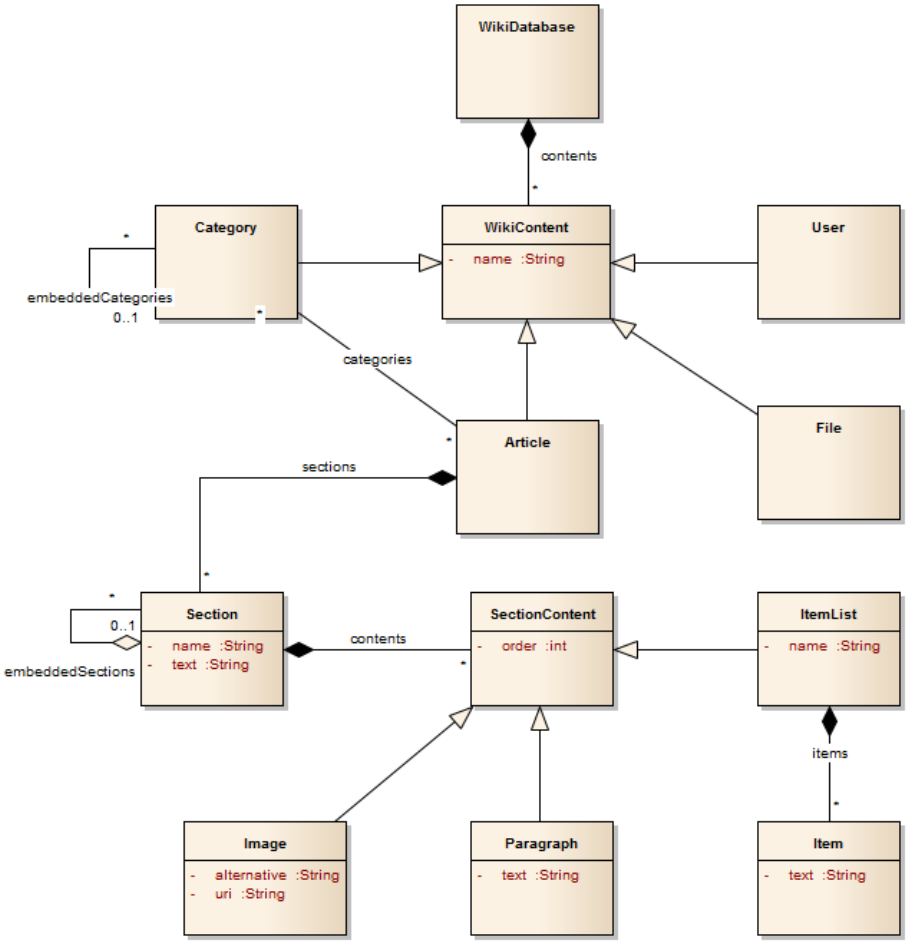

Figura 6.6: Modelo de herramientas wiki

Modelo de herramientas de modelado visual

En Ingeniería del Software, así como en otras disciplinas, es habitual la utilización de herramientas de edición de diagramas mediante algún lenguaje visual. En particular, las herramientas de edición UML permiten construir modelos de software, de tal forma que permiten gestionar gran parte de los productos de trabajo de un proyecto.

A partir del análisis de varias herramientas UML como *Enterprise Architect*, *Visual Paradigm for UML* o *Rational Rose* se ha diseñado el modelo *Visual Modeling tool Model* (VMM) que se muestra en la figura 6.7. Este modelo permite representar la estructura básica de estas herramientas UML, aunque no excluye aquellas otras que permiten utilizar otros lenguajes visuales para el modelado de sistemas software u otras entidades. A continuación, se presentan los elementos de este modelo.

- *ModelRepository*. Clase contenedora de todos los proyectos gestionados con la herramienta de modelado.
- *Project*. Clase contenedora de todos los modelos generados para un determinado proyecto.
- *Package*. Clase que representa a un paquete de modelos con el cual organizar los diferentes modelos.
- *Diagram*. Clase que representa a un determinado diagrama modelado con algún lenguaje visual.
- *DiagramType*. Enumeración con los posibles tipos de diagramas reconocidos por la herramienta de modelado, como por ejemplo los diagramas de clases o de estados de UML.
- *Element*. Clase que representa a un elemento que forma parte de algún diagrama.
- *ElementType*. Enumeración con los posibles tipos de elementos que pueden participar en un diagrama. Los casos de uso, los actores o los estados son ejemplos de tipos de elementos en modelos UML.
- *Connector*. Clase que representa a una determinada relación entre dos elementos de modelado.

- *ConnectorType*. Enumeración con los posibles tipos de conectores que pueden vincular dos elementos, como por ejemplo las asociaciones, generalizaciones y relaciones de inclusión o extensión de UML.

Modelo de herramientas de gestión de tareas

A partir del estudio de las características de las herramientas de gestión de tareas o *issue tracking systems*, se ha elaborado el modelo *Issue Tracking tool Model* (ITM) que se muestra en la figura 6.8. Este modelo incluye los siguientes elementos:

- *IssueTrackingDatabase*. Clase contenedora de todos los proyectos software gestionados en la herramienta.
- *Project*. Clase que representa a un proyecto software controlado desde la herramienta.
- *User*. Clase que representa a un usuario registrado en la herramienta.
- *Role*. Clase que representa el rol que juegan los usuarios en los proyectos.
- *Member*. Clase utilizada para asociar un determinado usuario a un proyecto, haciendo uso de un rol determinado.
- *Issue*. Clase que describe una tarea o unidad concreta de trabajo necesaria para evolucionar un sistema informático, añadiendo una nueva característica, solucionando una incidencia, etc.
- *IssueCategory*. Clase que permite realizar una clasificación específica de las tareas de un determinado proyecto.
- *IssueStatus*. Enumeración con los diferentes estados por lo que puede pasar una determinada tarea.
- *IssuePriority*. Enumeración con los diferentes niveles de prioridad que puede tomar una tarea.
- *IssueDependency*. Clase de asociación que permite representar la dependencia entre dos tareas concretas.
- *DependencyType*. Enumeración con los diferentes tipos de dependencias entre tareas.

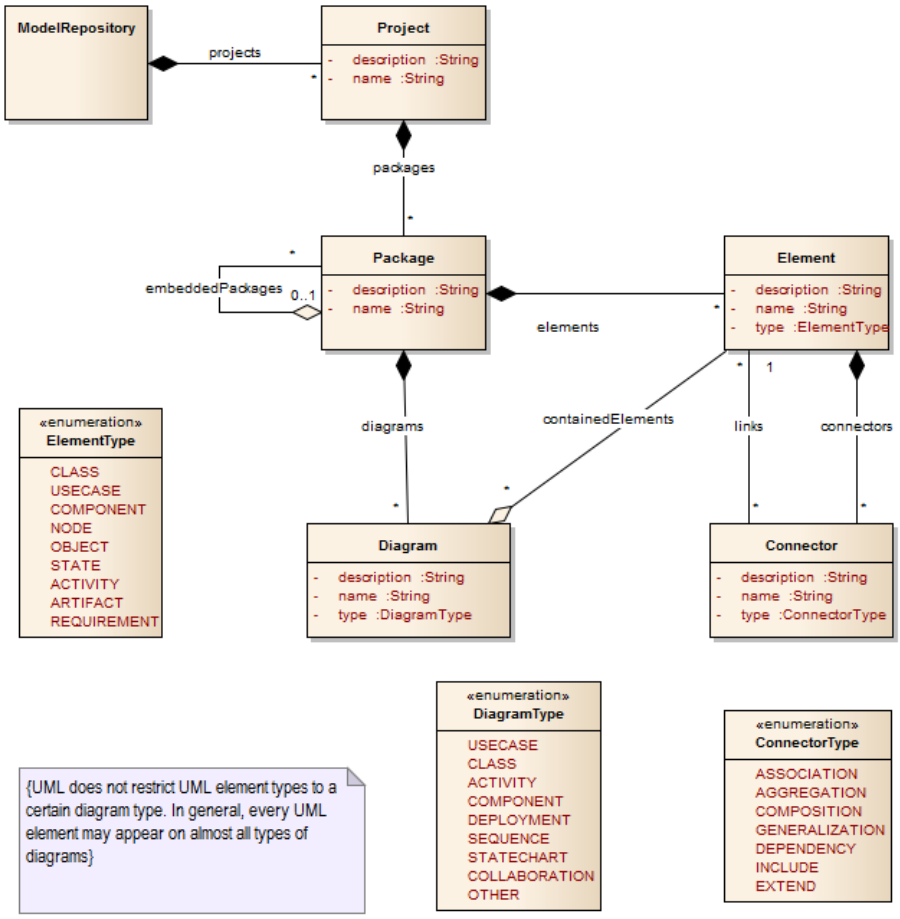

Figura 6.7: Modelo de herramientas de modelado visual

- *Tracker*. Clase utilizada para identificar el listado de tareas de un determinado tipo, típicamente *bugs* o *features*.
- *Version*. Clase que describe cada una de las versiones del software objeto del proyecto gestionado en la herramienta.
- *VersionStatus*. Enumeración con los diferentes estados por lo que puede pasar una determinada versión del software.

6.2.3. Modelos de herramientas específicas

Cada vez son más las herramientas de soporte o plataformas integradas que aglutinan varias características para cubrir diferentes aspectos del proceso software. Por ejemplo, la herramienta *Redmine*, aunque está dirigida fundamentalmente a la gestión de tareas, incluye también un control de versiones de código, una wiki, un sistema de foros, noticias y contenidos dinámicos.

Estos modelos PSM se forman a partir de la composición de los modelos de herramientas genéricas para cubrir las múltiples capacidades que puedan ofrecer las herramientas de soporte. De esa manera, los modelos PSM de sistemas genéricos podrán ser reutilizados para componer los modelos PSM finales de herramientas concretas. A continuación, se presentan los modelos resultantes de componer los modelos de la sección anterior para tres herramientas específicas.

Modelo de MediaWiki

MediaWiki es un software escrito en *PHP Hypertext Preprocessor* (PHP) para implementar sistemas wiki de edición colaborativa. Originariamente fue desarrollado para su utilización en *Wikipedia*, aunque al ser software libre puede utilizarse para otros fines. Uno de los posibles usos es como herramienta de apoyo al desarrollo de software.

El modelo de *MediaWiki* se compone únicamente del modelo WIKIM ya que esta herramienta sólo implementa las facilidades propias de los sistemas de edición wiki. Lo vemos en la figura 6.9.

Modelo de Enterprise Architect

Enterprise Architect es una herramienta de modelado visual basada principalmente en el estándar UML. Esta herramienta ofrece soporte al diseño de sistemas software, modelado de procesos de negocio, simulación y otras capacidades adicionales.

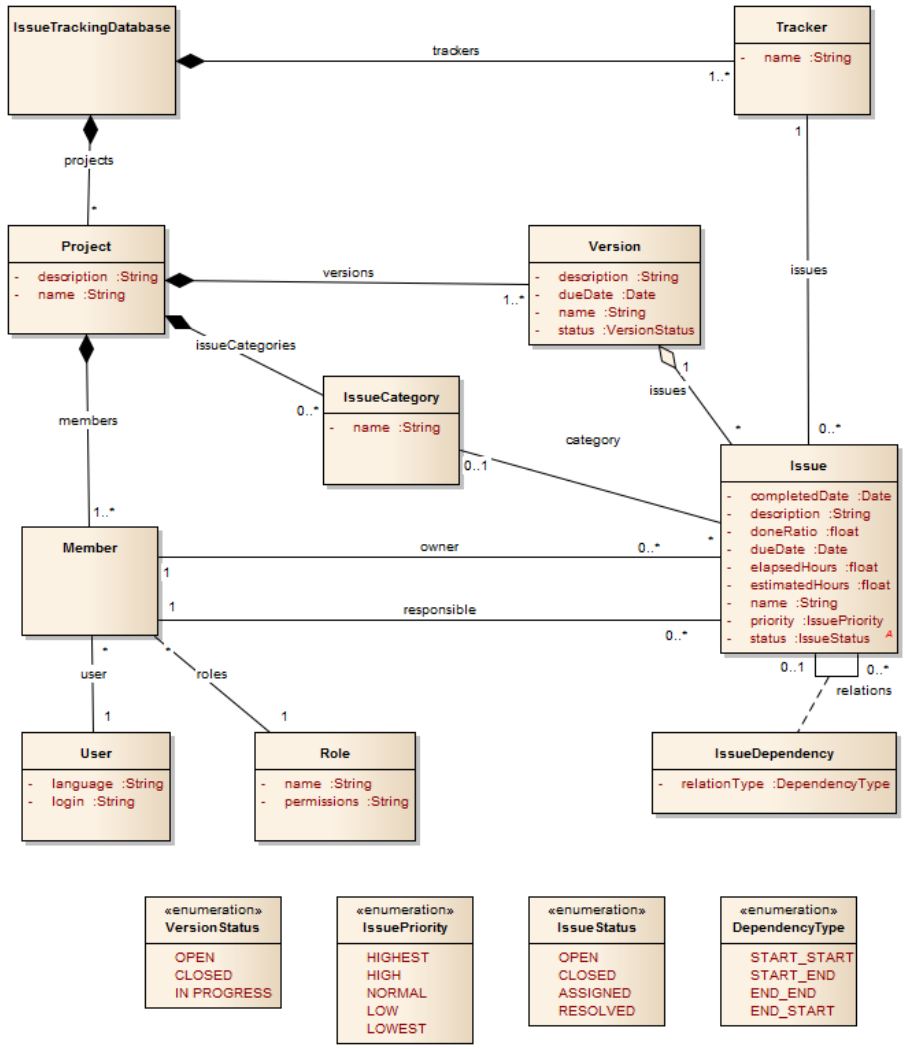

Figura 6.8: Modelo de herramientas de gestión de tareas

Figura 6.9: Modelo de MediaWiki

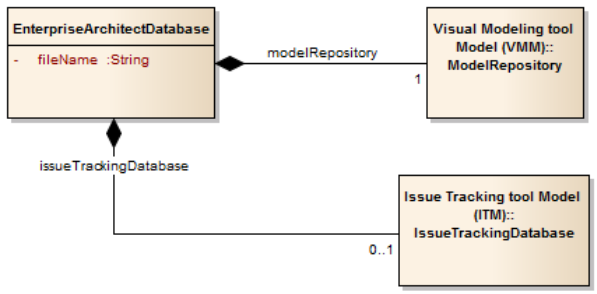

Figura 6.10: Modelo de Enterprise Architect

Además de las características habituales en las herramientas UML, como son el diseño y almacenamiento de los diferentes tipos de modelos UML, *Enterprise Architect* incorpora otras características adicionales relativas a la gestión de proyectos. Por ello, el modelo de *Enterprise Architect* (figura 6.10) se compone a partir de los modelos propios de los sistemas de gestión de tareas (ITM) y de las herramientas de modelado visual (VMM).

Modelo de Redmine

Redmine es una aplicación web para la gestión de proyectos desarrollada con el *framework Ruby on Rails*. Esta herramienta cumple con las características básicas que ofrecen las herramienta de gestión de tareas. Además de esto, *Redmine* ofrece a los usuarios una wiki y otros servicios adicionales como foros, gestión de contenidos y visualización de repositorios externo de código fuente, etc. En

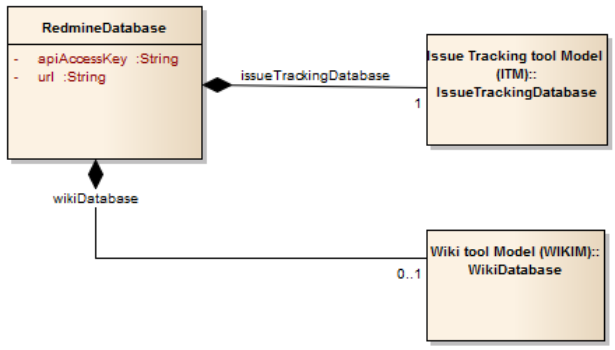

Figura 6.11: Modelo de Redmine

la figura 6.11 podemos ver el modelo resultante para esta herramienta.

6.3. Relaciones entre modelos

Existe una estrecha relación entre cada uno de los modelos presentados en la sección anterior. Los elementos de los modelos de procesos software SPEM están asociados con elementos de los modelos de despliegue SWPM y SPCM. Éstos lo hacen a su vez con elementos de los modelos de herramientas genéricas WIKIM, VMM e ITM. En última instancia, los modelos de herramientas específicas utilizan los modelos definidos en el nivel justo anterior. En la figura 6.12 podemos comprobar dichas relaciones.

La implementación de las relaciones depende, al igual que como sucede con los modelos, del espacio tecnológico empleado. En MDE, las asociaciones estereotipadas con *trace* se traducen en reglas de transformación ATL, mientras que en LOD se materializan en axiomas de equivalencia o especialización y en reglas de razonamiento semántico. Con respecto a las asociaciones marcadas con *import*, en el espacio MDE representan la mera reutilización de elementos de los metamodelos, mientras que en LOD no tienen correspondencia dado que para este enfoque no se han implementado vocabularios específicos para las herramientas. En la sección 6.4 se ofrecen más detalles sobre la implementación de las relaciones entre modelos.

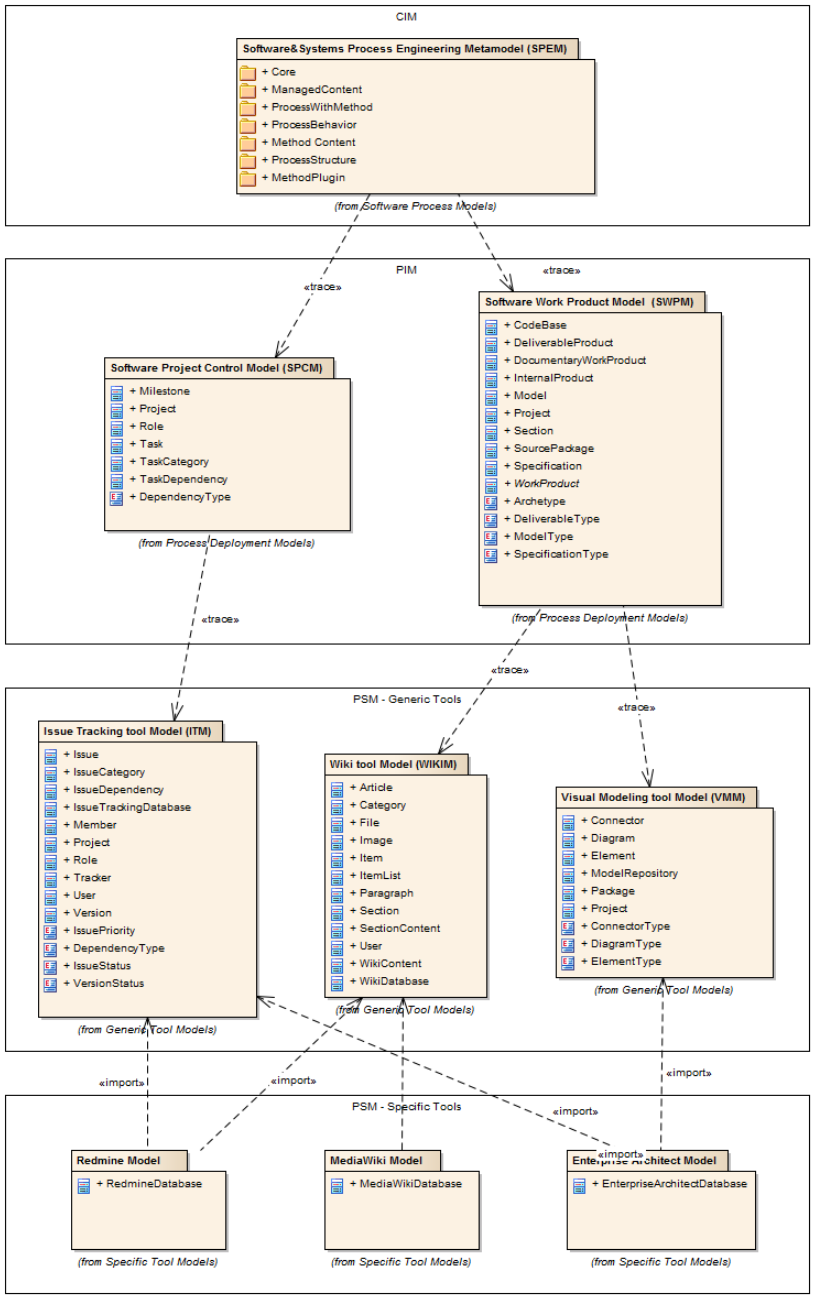

Figura 6.12: Relaciones existentes entre los distintos niveles de modelado

6.3.1. Relaciones entre modelos de procesos y modelos de despliegue

A continuación, se muestran las relaciones entre los modelos de procesos software y los diferentes modelos de despliegue.

Relaciones entre el modelo de productos de trabajo y el modelo de proceso

En la figura 6.13 podemos observar la relación entre los elementos del modelo de proceso SPEM y los elementos del modelo de productos de trabajo software SWPM. La justificación de estas relaciones se incluyen en la tabla 6.1.

A la hora de implementar las reglas de transformación necesarias para obtener un modelo SWPM a partir de un modelo SPEM, se necesitan tomar ciertas decisiones, como por ejemplo, si un dominio hace referencia a un producto documental o una base de código, o si un artefacto determinado es un modelo, una especificación o un paquete de código fuente. Todas estas decisiones deben ser tomadas por el Ingeniero de Procesos y se plasmarán en el modelo SWPM resultante.

Relaciones entre el modelo de control de proyecto y el modelo de proceso

En la tabla 6.2 se describen las relaciones existentes entre los elementos del modelo de SPEM y los elementos del modelo de control de proyectos software SPCM. La figura 6.14 ilustra dichas relaciones.

6.3.2. Relaciones entre modelos de despliegue y modelos de herramientas genéricas

En este apartado se presentan las relaciones entre los distintos modelos de despliegue y los modelos de herramientas genéricas.

Relaciones entre el modelo wiki y el modelo de productos de trabajo

En la figura 6.15 podemos observar la relación entre los elementos del modelo de productos de trabajo y los elementos del modelo de un sistema wiki. La justificación de estas relaciones se incluyen en la tabla 6.3.

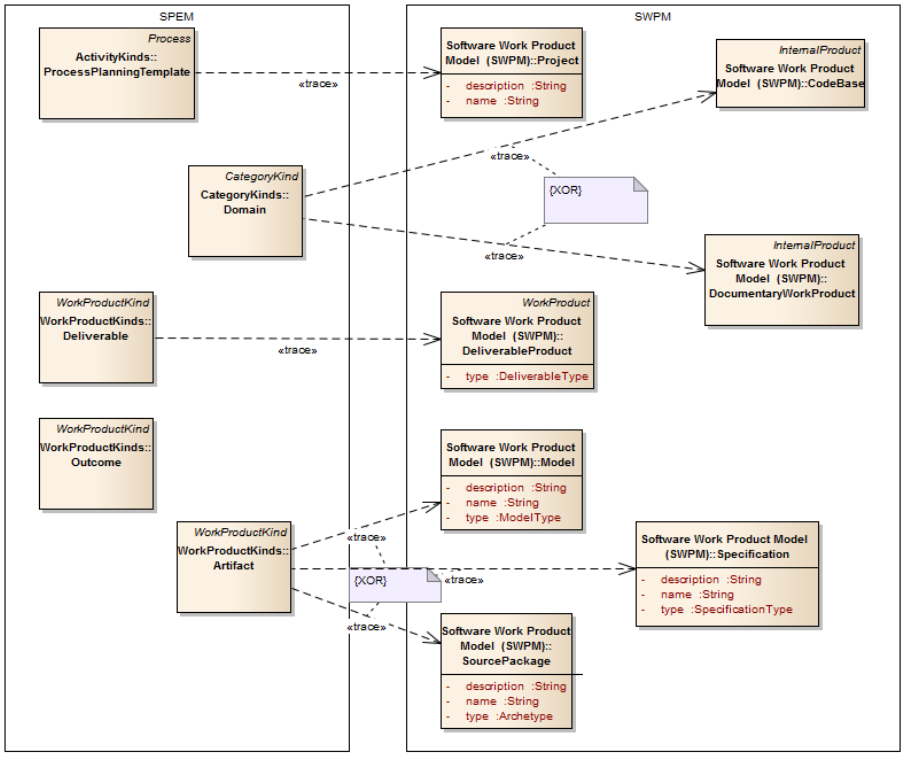

Figura 6.13: Relaciones entre los modelos de SPEM y SWPM

Tabla 6.1: Descripción de las relaciones entre SPEM y SWPM

SPEM	SWPM	Justificación
Process Planning Template	*Project*	Un proceso plantilla de planificación SPEM se materializa en un proyecto software.
Domain	*Documentary Work Product* o *Base Code*	Según la especificación de SPEM, el dominio es una agrupación lógica de productos de trabajo que presentan cierta similitud. Por ejemplo, en *OpenUP*, se establecen Requisitos, Arquitectura y Test, entre otros dominios. Por ello, cada dominio se materializará en un documento de trabajo o bien, en una base de código.
Deliverable	*Deliverable Product*	Los productos de trabajo clasificados como *Deliverable* en SPEM se asocian con el concepto *Deliverable Product*, respetando su misma semántica pero enriqueciéndola con el tipo de entregable.
Outcome	-	Dado que el *Outcome* en SPEM se refiere a un producto no tangible y se emplea para indicar el resultado o el estado de un producto, no tendrá representación directa en SWPM.
Artifact	*Model*, *Specification* o *Source Package*.	Los artefactos en SPEM son materializados en SWPM según la naturaleza del mismo, existiendo tres tipos diferenciados: modelos, especificaciones no basadas en modelos y en código fuente.

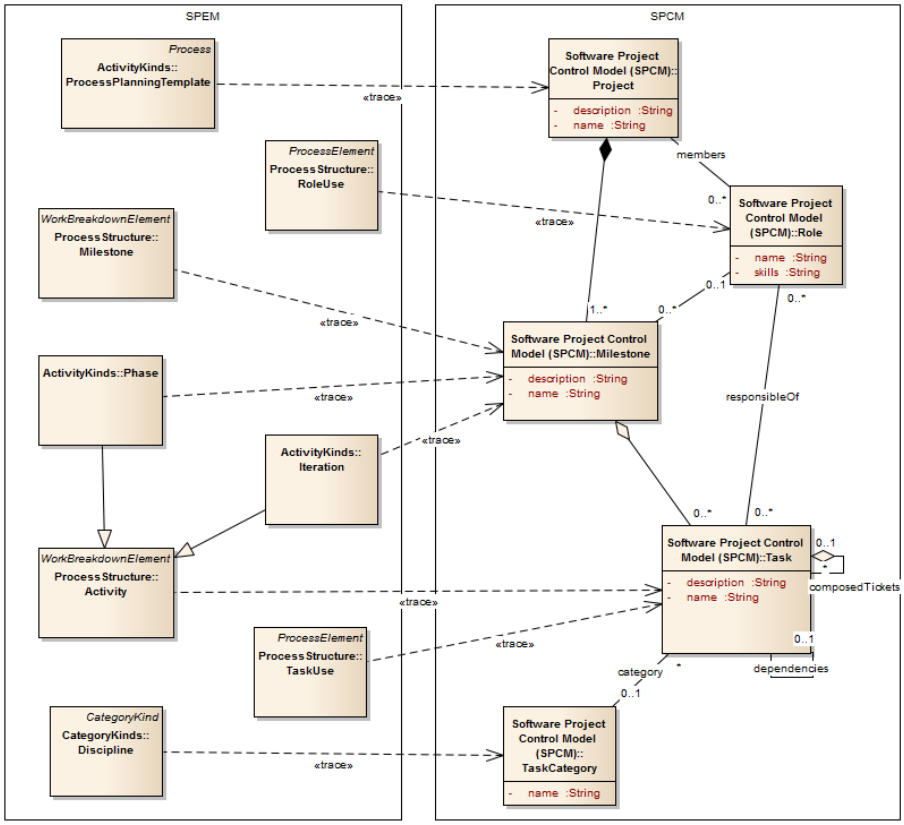

Figura 6.14: Relaciones entre los modelos de SPEM y SPCM

Tabla 6.2: Descripción de las relaciones entre SPEM y SPCM

SPEM	SPCM	Justificación
Process Planning Template	*Project*	Un proceso de planificación SPEM se materializa en un proyecto software.
RoleUse	*Role*	La participación de un rol en el proceso software se sintetiza como un rol del modelo SPCM.
Milestone	*Milestone*	El concepto de *milestone* se mantiene en el modelo de control de proyectos software.
Phase	*Milestone*	Una fase en SPEM representa un periodo de tiempo donde realizan actividades, lo cual se concretará en un nuevo *milestone*.
Iteration	*Milestone*	Al igual que el anterior, una iteración en SPEM representa un periodo de tiempo donde realizan actividades, pero que pueden repetirse más de una vez. En este caso, cada iteración se materializaría en un *milestone* diferente.
Activity	*Task*	Una actividad en SPEM es un elemento de desglose de trabajo que puede contener a su vez otras actividades. La actividad puede especializarse en un proceso, fase o iteración. Las actividades no especializadas podrán ser materializadas en tareas contenedoras de otras.
TaskUse	Task	El uso de tareas en SPEM representa la unidad más pequeña de trabajo, las cuales serán materializadas en tareas del modelo SPCM.
Discipline	*TaskCategoy*	Las disciplinas en el modelo de proceso permitirán categorizar los tareas asociadas al proyecto.

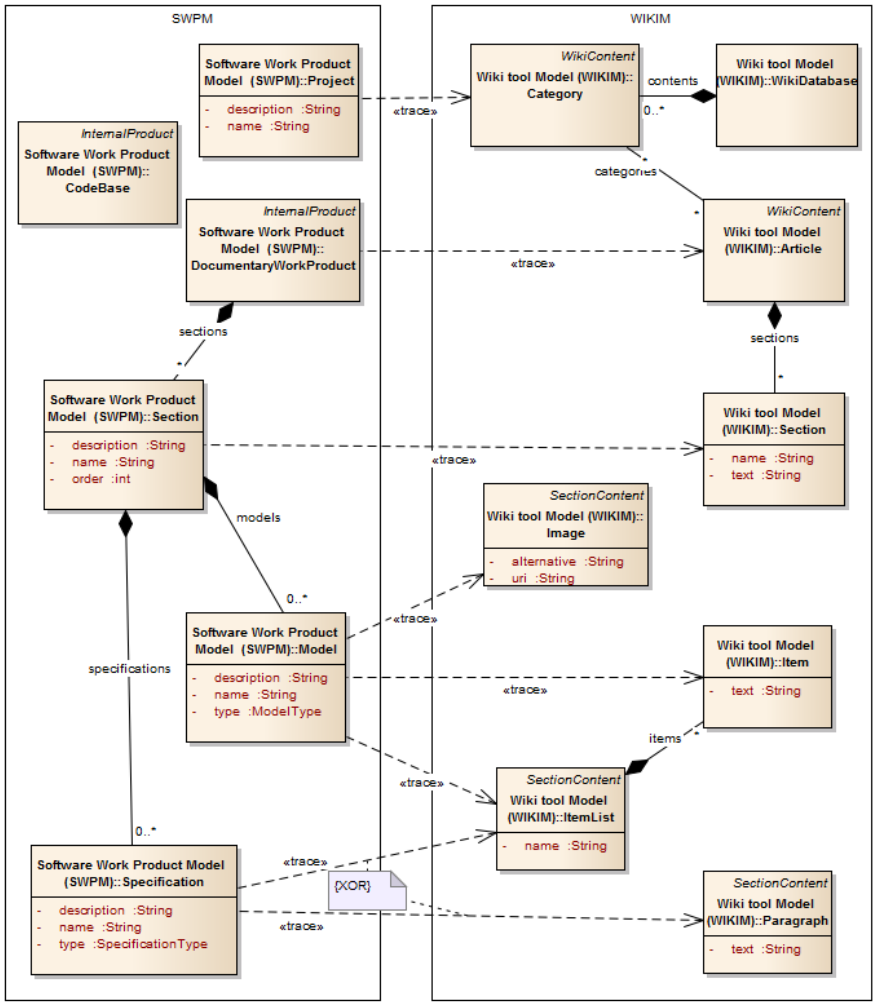

Figura 6.15: Relaciones entre los modelos de SWPM y WIKIM

Relaciones entre el modelo de herramientas de modelado visual y el modelo de productos de trabajo

En la figura 6.16 podemos observar la relación entre los elementos del modelo de productos de trabajo y los elementos del modelo de un sistema de edición de modelos visuales. La justificación de estas relaciones se incluyen en la tabla 6.4.

Relaciones entre el modelo de sistemas de gestión de tareas y el modelo de control de proyectos

La relación entre los elementos del modelo de control de proyectos y los elementos del modelo de sistemas de gestión de tareas se presentan en la figura 6.17. Como podemos apreciar, ambos modelos son similares y comparten muchos de sus elementos. Esto es debido a que los sistemas de gestión de tareas se han diseñado con el objetivo de facilitar la monitorización de los proyectos, en detrimento de los sistemas estáticos de planificación Gantt o de las hojas de calculo creadas ad-hoc para la gestión. La justificación de estas relaciones se incluyen en la tabla 6.5.

6.4. Herramientas de apoyo

El método para el despliegue y evaluación de procesos descrito en esta tesis consta de las siguientes actividades: modelado de procesos software, adaptación de las herramientas de soporte, apertura de las herramientas y desarrollo de soluciones de integración para la evaluación de los procesos. En esta sección se describen los componentes que se han desarrollado para la adaptación y apertura de las herramientas.

6.4.1. Componentes para la adaptación de herramientas de soporte

En esta sección se describe cómo se han implementado los modelos y las relaciones en el espacio tecnológico de MDE y las herramientas de apoyo.

Implementación de los metamodelos y las reglas de transformación

Con respecto a los modelos de procesos software (nivel CIM) se utiliza UMA, la implementación de SPEM por parte de la comunidad *Eclipse*. Este metamodelo se basa en el meta-metamodelo *Ecore*. El resto de modelos planteados en

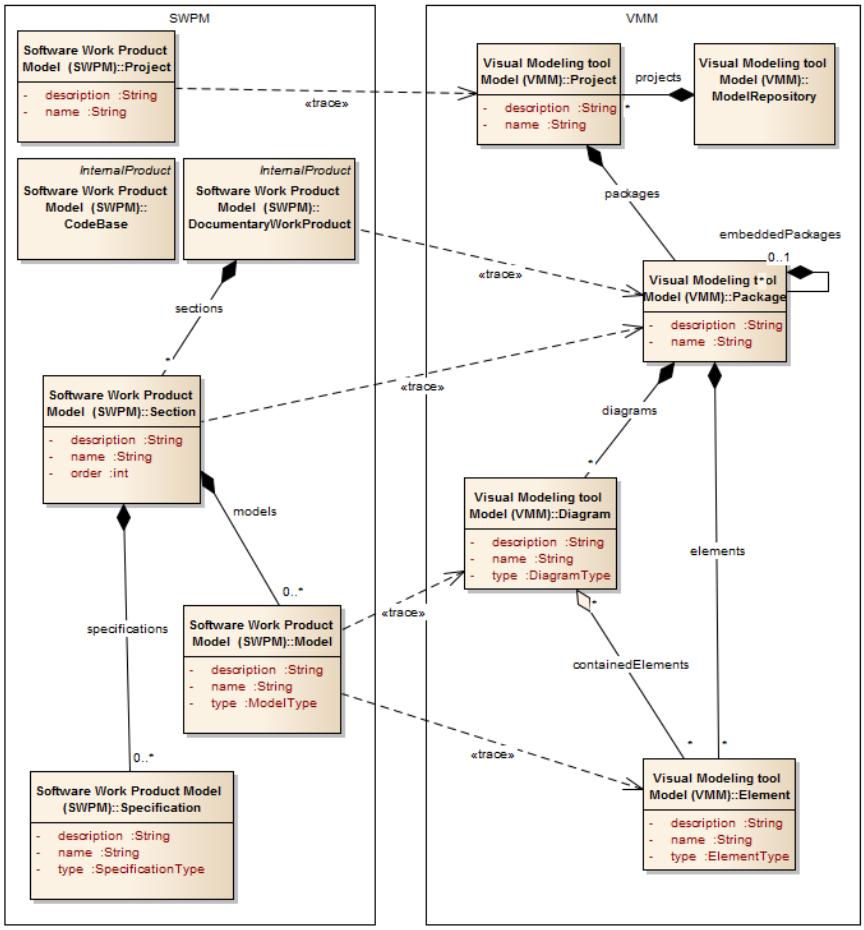

Figura 6.16: Relaciones entre los modelos de SWPM y VMM

Tabla 6.3: Descripción de las relaciones entre SWPM y WIKIM

SWPM	WIKIM	Justificación
Project	*Category*	Con el objetivo de poder clasificar todos los artículos wiki correspondientes a un proyecto, se generará una categoría para cada proyecto.
Documentary Work Product	*Article*	Los productos de trabajo de tipo documental serán materializados en artículos de la wiki.
CodeBase	–	Los productos *CodeBase* de SWPM quedan al margen de estas relaciones, dado que una wiki no es un software diseñado para gestionar código fuente.
Section	*Section*	Las secciones de los documentos de SWPM se materializan en secciones de los artículos wiki.
Model	*Image*, *ItemList* e *Item*	*MediaWiki* no soporta la edición de modelos, aunque sí dispone de algunos elementos con los que suplir, al menos, su visualización. Por ello, los modelos se concretarán en una imagen incluida en el artículo junto con una lista de ítems para los elementos del modelo, como por ejemplo, las clases de un modelo conceptual.
Specification	*Paragraph*	Las especificaciones no descritas mediante modelos, podrán concretarse como párrafos de un artículo.

Tabla 6.4: Descripción de las relaciones entre SWPM y VMM

SWPM	VMM	Justificación
Project	*Project*	Los proyectos en SWPM se materializan en proyectos dentro de la herramienta de modelado.
CodeBase	–	Los productos *CodeBase* de SWPM quedan también al margen de estas relaciones, dado que una herramienta de edición de modelos no está orientada a gestionar código fuente.
Documentary Work Product	*Package*	Los productos de trabajo de tipo documental serán materializados en paquetes de primer nivel.
Section	*Package*	Las secciones de los documentos de SWPM se materializan en paquetes de modelos.
Model	*Diagram* y *Element*	Por cada modelo contemplado en SWPM, se generará el diagrama correspondiente, incluyendo un elemento de ejemplo.
Specification	–	Las especificaciones no descritas mediante modelos no serán materializadas en elementos de glsVMM.

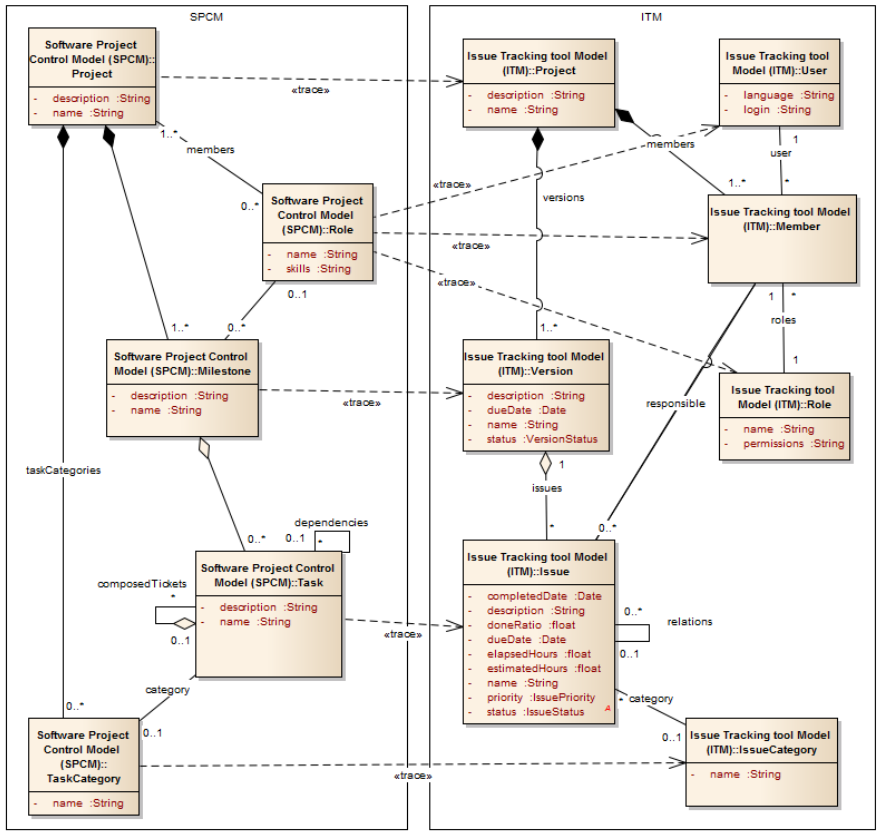

Figura 6.17: Relaciones entre los modelos de SPCM e ITM

Tabla 6.5: Descripción de las relaciones entre SPCM e ITM

SWPM	ITM	Justificación
Project	*Project*	Los proyectos existentes en SPCM serán vinculados con proyectos registrados en la herramienta.
Role	*Member*, *User* y *Role*	Un miembro de proyecto definido en SPCM corresponderá con un usuario de la herramienta con un determinado rol y asociado al proyecto en cuestión.
Milestone	*Version*	Los *milestones* del modelo de control de proyectos se materializan en versiones del proyecto en la herramienta de gestión.
Task	*Issue*	Las tareas en SPCM se concretan en *issues* vinculados al *tracker* por defecto del proyecto.
TaskCategory	*IssueCategory*	Las categorías de las tareas corresponderán con las categorías de los *issues* dentro de la herramienta.

esta tesis, tanto de nivel PIM como de los niveles PSM, fueron implementados utilizando la misma tecnología que UMA, esto es, con el lenguaje de metamodelado *Ecore*. A la hora de implementar los metamodelos *Ecore* para los modelos conceptuales descritos la sección 6.2, es preciso realizar algunos cambios en su estructura ya que mientras que en UML se utiliza una estructura de tipo grafo, en *Ecore* se utiliza una estructura de tipo árbol, por lo cual, todas las metaclases deben estar contenidas directa o indirectamente en una metaclase principal. Además, fue preciso definir la navegabilidad necesaria para cada una de las asociaciones presentes en el modelo.

Con el objetivo de derivar automáticamente los modelos de despliegue (PIM) a partir de los modelos de proceso (CIM), se ha implementado un conjunto de reglas de transformación con el lenguaje ATL. Del mismo modo, se ha utilizado este lenguaje para la implementación de las reglas de transformación entre los modelos de despliegue (PIM) y los modelos de herramientas (PSM).

Es importante destacar que no se han implementado reglas de transformación para todas y cada una de las relaciones existentes entre los elementos de los metamodelos. Esto es así debido a que se necesita la intervención del Ingeniero de Procesos para tomar ciertas decisiones. Por ejemplo, no hay manera de discernir directamente si un determinado *WorkProductDefinition* en SPEM (o en su implementación en UMA) corresponde a un *Model* o a una *Specification* de SWPM, o bien, si se tratase de un modelo, decidir cuál es su tipo. Por ello, a la hora de implementar las relaciones, es preciso tomar ciertas decisiones por

defecto, siguiendo de este modo el principio de *Convention over configuration* [31]. En el caso particular de la transformación de UMA a SWPM se ha tomado la decisión de generar modelos de clases UML para todos los artefactos en UMA. Podemos observar cómo se ha implementado la regla de transformación correspondiente a esta casuística en el listado de código 6.1.

```
--- Matching a UMA Artifact with a SWPM Model by convention
rule Artifact2Model {
 from
  art: UMA!Artifact (
   art.presentationName.size() > 0
  )
 to
  model: SWPM!Model (
   name <- art.presentationName,
   description <- art.briefDescription,
   type <- #LOGICAL
  )
}
```

Listado 6.1: Definicion de una regla de transformación en ATL

En el anexo B se incluyen las implementaciones con *Ecore* de los modelos relacionados con los productos de trabajo software (SWPM, WIKIM y VMM) y con sus herramientas de soporte, así como las reglas ATL necesarias para las transformaciones entre los modelos.

Herramientas para el despliegue de procesos software

Para el diseño de los modelos de procesos software se hará uso del entorno EPF, una versión adaptada del entorno *Eclipse* que ofrece un completo editor para la definición de metodologías y procesos de software. Los modelos diseñados con esta herramienta se almacenan en ficheros XMI conformes al metamodelo UMA.

Para la adaptación de las herramientas de soporte se ha implementado un software específico para tal fin. Este software, distribuido en forma de *plugins* para *Eclipse*, dispone de una serie de utilidades para editar y transformar los modelos implicados en el despliegue de los procesos. El software, denominado *Software Process Deployment Toolkit* (SPDT), se basa en tecnologías *Java* y se ha construido haciendo uso de las tecnologías incluidas dentro de *Eclipse Modeling Project*. En la figura 6.18 se muestra la arquitectura de este software, incluyendo los paquetes y los componentes principales.

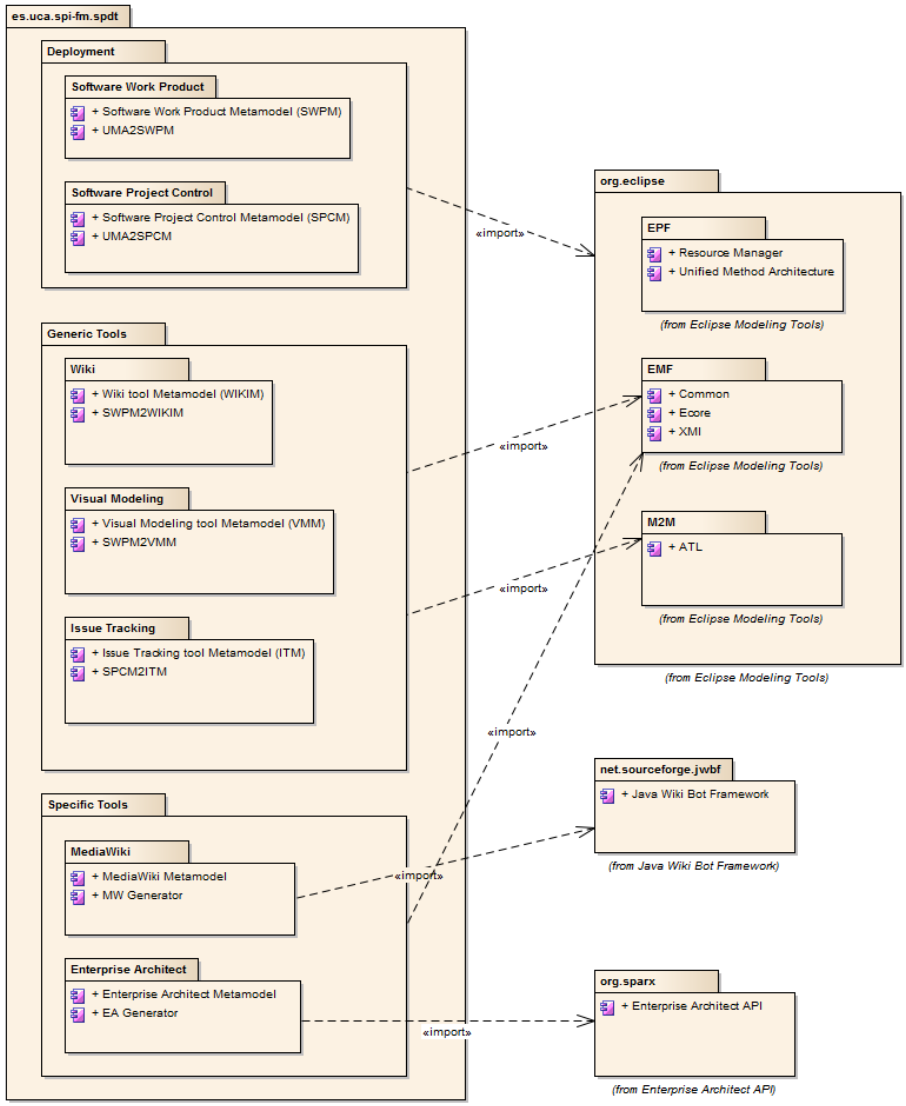

Figura 6.18: Diagrama de componentes de SPDT

Las tareas requeridas para adaptar las herramientas de soporte son las siguientes, según se explica en el apartado 6.1.2: (i) Derivar modelos de despliegue del proceso software, (ii) Refinar modelos de despliegue del proceso software, (iii) Derivar modelos de herramientas genéricas, (iv) Componer modelos de herramientas específicas, (v) Refinar modelos de herramientas específicas y (vi) Configurar las herramientas de soporte. Algunas de estas tareas son manuales, mientras que otras pueden realizarse de forma automática.

Para derivar los modelos de despliegue se emplea ATL que, además de proporcionar el lenguaje para el diseño de las reglas de transformación, dispone de un motor de ejecución para la transformación de los modelos M2M.

Para las tareas de refinamiento de modelos, se ha dispuesto de una serie de editores reflexivos (tipo árbol) para todos los modelos planteados en el *framework*. Con estos editores, el Ingeniero de Procesos podrá personalizar los modelos generados automáticamente para que estos se adapten a las necesidades de su organización.

La derivación de los modelos de herramientas genéricas y la composición de modelos de herramientas específicas se realizarán de forma automática mediante la ejecución de otras reglas con el motor ATL. Posteriormente, el Ingeniero de Sistemas podrá refinar los modelos resultantes para ajustar ciertos parámetros de las herramientas específicas, como por ejemplo las credenciales de acceso.

Para configurar las herramientas de soporte, el Ingeniero de Sistemas deberá lanzar unos scripts específicos. La implementación de estos scripts depende de los mecanismos que ofrezca la herramienta en cuestión para manipular programáticamente la información que gestiona. En la solución técnica implementada hasta el momento, se han desarrollado los artefactos necesarios para desplegar sobre *MediaWiki* y *Enterprise Architect*. En el caso de *Enterprise Architect*, se ha construido un software *Java* que a partir de un modelo conforme a su metamodelo, se genera automáticamente un proyecto de ejemplo conteniendo los paquetes, los diagramas UML y los elementos definidos en el modelo de entrada. Para ello, se utiliza el API que distribuye el propio fabricante. En el caso de *MediaWiki*, se ha utilizado otra API que permite conectarse a una instalación remota de la herramienta y crear o modificar artículos sobre la misma.

6.4.2. Componentes para la apertura de herramientas de soporte

En esta sección se detallan los vocabularios y los mecanismos software que se han implementado para la apertura de los datos gestionados por las herramientas de soporte a la gestión o producción de software.

Implementación de los vocabularios y las reglas de inferencia

Para asegurar que las soluciones de integración sean lo más independientes posible de las herramientas concretas que se desean integrar es preciso utilizar vocabularios compartidos. Una premisa fundamental en el enfoque LOD es que a la hora de publicar datos en RDF es conveniente reutilizar, siempre que sea posible, los términos procedentes de vocabularios ya existentes, en lugar de reinventarlos. De esa forma, se aumenta la probabilidad de que las aplicaciones preparadas para los vocabularios ya existentes puedan consumir los nuevos datos sin necesidad de realizar modificaciones adicionales [75].

Con el objetivo de publicar la información relativa a los productos de trabajo y a ciertos aspectos del seguimiento de los proyectos, se propone la utilización de un conjunto de vocabularios LOD adecuados para tal fin. Sin embargo, en el catálogo[5] de facto de vocabularios *RDF Schema* apenas se han encontrado vocabularios que permitan cubrir los conceptos y las relaciones existentes en los modelos planteados en esta tesis.

Como punto de partida se propone utilizar el vocabulario DOAP, con el cual se pueden definir los datos básicos de los proyectos gestionados bajo forjas o repositorios de software. Sin embargo, dado que este vocabulario no incluye muchos de los aspectos del proceso software, como por ejemplo las versiones o las tareas, se hace necesario implementar nuevos vocabularios para los modelos de despliegue (SWPM y SPCM) y para los modelos de herramientas genéricas (VMM, WIKIM e ITM). Estos vocabularios fueron implementados en *RDF Schema*.

Posteriormente, es necesario implementar las relaciones existentes entre los modelos de los niveles PIM y PSM, además de las relaciones con otros términos de vocabularios externos, para facilitar así la futura integración con terceros sistemas. Para ello, se emplearon las clausulas de equivalencia y de especialización *owl:equivalentClass*, *owl:equivalentProperty*, *rdfs: subClassOf* y *rdfs:subPropertyOf*, que ofrece el enfoque LOD y su paradigma matriz, la Web Semántica.

En el listado 6.2 podemos visualizar un ejemplo de axioma donde se describe que la entidad *Project* en el contexto de una herramienta de modelado visual es una especialización del concepto *Project* como contenedor abstracto de todos los productos de trabajo generados durante el desarrollo o mantenimiento de un determinado software.

[5] http://lov.okfn.org/dataset/lov/

```
@prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vmm:
  <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#> .
@prefix swpm:
  <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#> .

vmm:Project rdfs:subClassOf swpm:Project .
```

Listado 6.2: Definición de una axioma de especialización en RDF

Sin embargo, no siempre existe una correspondencia univoca entre los elementos de los modelos de niveles diferentes. Por ejemplo, un *Package* en el modelo VMM puede ser una materialización de un *DocumentaryWorkProduct* o de un *Section* en el modelo SWPM. Para modelar este tipo de relaciones, necesitamos utilizar un razonador semántico o motor de reglas que permita inferir consecuencias lógicas a partir de una serie de hechos descritos como tripletas RDF. En el listado 6.3 podemos ver un ejemplo de una regla de derivación.

```
@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix vmm:
  <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#> .
@prefix swpm:
  <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#> .

# if the resource ?1 is a Project
  (?1 rdf:type vmm:Project),
# and the resource ?1 has a package ?2
  (?1 vmm:packages ?2)
# then the package ?2 is a Documentary WorkProduct
  (?2 rdf:type swpm:DocumentaryWorkProduct)
```

Listado 6.3: Definición de una regla de derivación en RDF

Los tipos enumerados existentes en los modelos conceptuales descritos en este framework se implementaron como instancias del vocabulario *Simple Knowledge Organization System* (SKOS), el estándar propuesto por la W3C para la definición de estructuras conceptuales, como tesauros, esquemas de clasificación, taxonomías y otros tipos de vocabularios controlados.

Finalmente, los vocabularios fueron publicados en la Web mediante la he-

rramienta *Neologism*, de forma que puedan ser consumidos por cualquier cliente RDF. En el anexo C se recogen las implementaciones con *RDF Schema* de los modelos relacionados con los productos de trabajo software (SWPM, WIKIM y VMM), los mappings con otros vocabularios y los esquemas de clasificación SKOS.

Herramientas para la apertura de datos

Es necesario implementar ciertos componentes para la exposición de los datos desde las herramientas de soporte al proceso software. De este modo, las herramientas dispondrán de interfaces que les permitan atender a peticiones HTTP sobre recursos particulares referenciados por su *Uniform Resource Identifier* (URI) y, opcionalmente, a peticiones de consultas vía SPARQL. Estas interfaces devolverán la información solicitada en alguno de los formatos de serialización de RDF.

Adaptador LOD de APIs de herramientas de gestión de tareas: La mayor parte de los sistemas de gestión de tareas basados en web ofrecen una interfaz desde donde poder obtener datos acerca de los proyectos, tareas, etc. Sin embargo, cada herramienta utiliza formatos diferentes para la publicación de los datos, como XML o *JavaScript Object Notation* (JSON). Además de la variedad de formatos, las herramientas definen modelos de datos propios y aunque son similares en contenido, los términos empleados son diferentes. Con el objetivo de proporcionar un único mecanismo de acceso y un formato común para los datos sobre proyectos de software gestionados bajo diferentes herramientas de gestión de tareas, se ha implementado un software para tal fin. *Abreforjas* es una aplicación escrita en *Django* que extrae y normaliza la información existente en diferentes forjas de software. De esta manera, con independencia de la fuente de procedencia utilizada, el acceso y el consumo de los datos será siempre idéntico. Esta aplicación permite cargar bajo demanda y de forma periódica los datos procedentes de las forjas y al mismo tiempo, enriquecer la información de los proyectos con una serie de métricas adicionales. Asimismo, la herramienta habilita una interfaz LOD para la publicación de RDF utilizando el vocabulario ITM. En la figura 6.19 podemos observar una captura de pantalla de la aplicación mostrando las métricas calculadas para un determinado proyecto.

Extensión para la publicación de LOD en aplicaciones web: Con el objetivo de acelerar el desarrollo de interfaces RDF para la publicación de LOD sobre aplicaciones construidas con *frameworks* web MVC, se han desarrollado

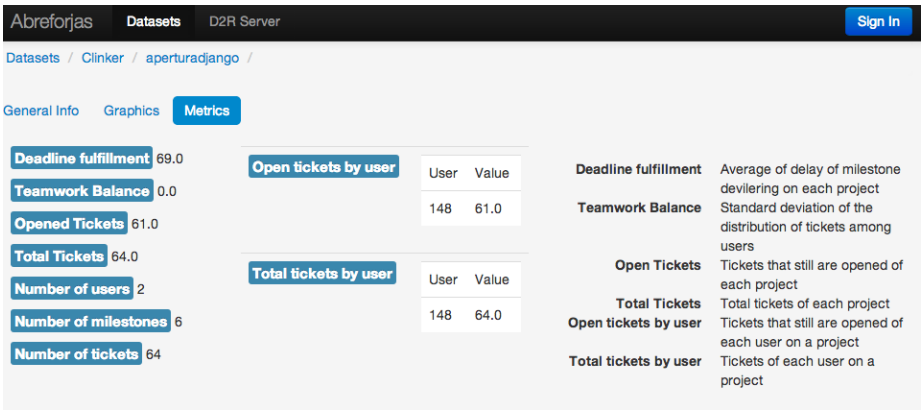

Figura 6.19: Interfaz de usuario de Abreforjas

dos herramientas: *EasyData/Rails*, destinada para aplicaciones construidas con el *framework Ruby on Rails* (RoR)[6] y *EasyData/Django*, destinada para aplicaciones basadas en Django[7], el *framework* de desarrollo web del lenguaje *Python*. Ambas utilidades permiten realizar un alineamiento entre los atributos y entidades de los modelos de las aplicaciones a extender y las propiedades y clases definidas en vocabularios RDF accesibles en internet. En la figura 6.20 podemos observar una captura de pantalla de *EasyData/Rails*, mostrando el *mapping* entre el modelo de una aplicación escrita con RoR, en este caso *Redmine* y un vocabulario externo, *Dublin Core* (DC). En la figura 6.21, observamos una pantalla similar en *EasyData/Django*.

Adaptador de datos para la publicación de LOD en Enterprise Architect: Enterprise Architect es una herramienta de edición UML que no ofrece libre acceso a su código fuente, aunque sí dispone de una API en varios lenguajes para consultar y manipular los datos de los proyectos. Dicha API ya se emplea para la generación automática de plantillas de proyectos a partir de un modelo de productos de trabajo (SWPM). Aunque dicha API podría emplearse para desarrollar un adaptador LOD, en esta ocasión se ha optado por crear un envoltorio directamente sobre la fuente de datos que utiliza. Para implementar

[6] http://rubyonrails.org/
[7] https://www.djangoproject.com/

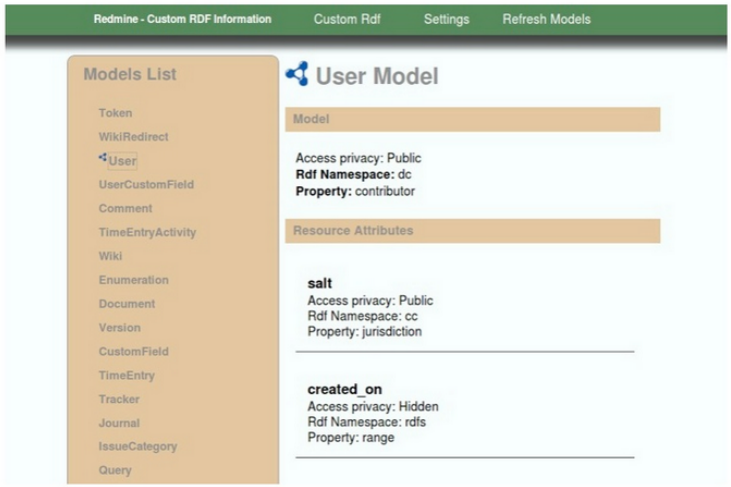

Figura 6.20: Interfaz de usuario de EasyData/Rails

Figura 6.21: Interfaz de usuario de EasyData/Django

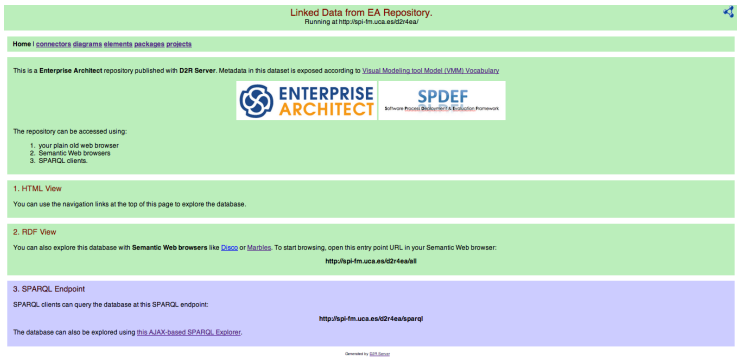

Figura 6.22: Interfaz de usuario de D2R Server

el adaptador se ha utilizado el software *D2R Server*[8] (véase figura 6.22) que permite publicar LOD directamente desde una base de datos relacional.

[8] http://d2rq.org

Capítulo 7

Evaluación del framework

En el capítulo anterior se ha presentado un marco de trabajo para el despliegue y evaluación de procesos software sobre herramientas de soporte. Este marco contempla el método para el despliegue y evaluación, una serie de modelos y relaciones, y un conjunto de herramientas de apoyo. La factibilidad técnica del método quedó demostrada mediante la implementación de esas herramientas.

En este capítulo se describe cómo se ha llevado a cabo la evaluación formal del *framework*, mediante la elaboración de dos casos de estudio y un ejemplo de un escenario descriptivo de uso. El primer caso de estudio consiste en el despliegue de una metodología de software sobre dos herramientas de soporte. El segundo es relativo a la evaluación de competencias en recursos de personal implicados en los procesos software. Por último, se presenta un escenario de integración con el cual se pueden realizar revisiones técnicas de calidad sobre proyectos de desarrollo software. En la tabla 7.1 se muestra la trazabilidad entre las hipótesis propuestas para ofrecer respuesta al problema, las actividades del método y herramientas que forman parte de la solución, y los casos de estudio y escenarios empleados para la evaluación del framework.

7.1. Despliegue de la metodología OpenUP

En esta sección se va a presentar el primero de los casos de estudio con el fin de asegurar la validez del *framework* propuesto. En este caso se describe cómo la metodología *OpenUP* puede desplegarse sobre determinadas herramientas de

Tabla 7.1: Trazabilidad entre los problemas, las soluciones y las evaluaciones

Problema	**Solución**		**Evaluación**	
Hipótesis	Actividad del método	Herramienta	Caso de estudio	Escenario descriptivo
H1	*Modelado de procesos*		*Despliegue de OpenUP*	
	Adaptación de herramientas	·*Software Process Deployment Toolkit*		
H2	*Apertura de herramientas*	·*Abreforjas* ·*EasyData/Rails* ·*EasyData/Django* ·*D2R para Enterprise Architect*	*Análisis de indicadores de personal*	*Revisiones de calidad*
	Desarrollo de soluciones de integración			

soporte al desarrollo de software, como *Enterprise Architect* y *MediaWiki*. En la figura 7.1 podemos comprobar este proceso.

La metodología de desarrollo *OpenUP*[1], donada por IBM a la *Fundación Eclipse*, es una versión ligera del Proceso Unificado [166]. Esta metodología sigue un enfoque iterativo e incremental y su ciclo de vida consta de cuatro fases: Inicio, Elaboración, Construcción y Transición.

La metodología se encuentra disponible en forma de una librería de métodos desarrollada con la herramienta EPF, por lo que puede reutilizarse su modelo de proceso. En la figura 7.2 podemos visualizar una captura de pantalla de la librería de métodos de *OpenUP* en la herramienta EPF.

A partir del modelo de proceso de *OpenUP* podemos derivar el modelo de productos de trabajo, resultado de ejecutar con el motor ATL las reglas de transformación definidas entre los metamodelos UMA y SWPM. En el modelo resultante, un proyecto que siga la metodología *OpenUP* contiene una serie de documentos, cada uno con una sección por defecto y una serie de artefactos clasificados como modelos UML.

En este momento, es necesario llevar a cabo un proceso de refinamiento

[1] http://epf.eclipse.org/wikis/openup/

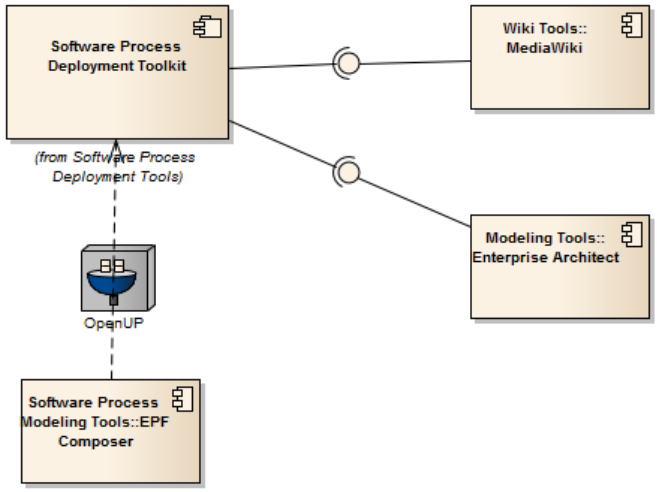

Figura 7.1: Proceso de despliegue de OpenUP

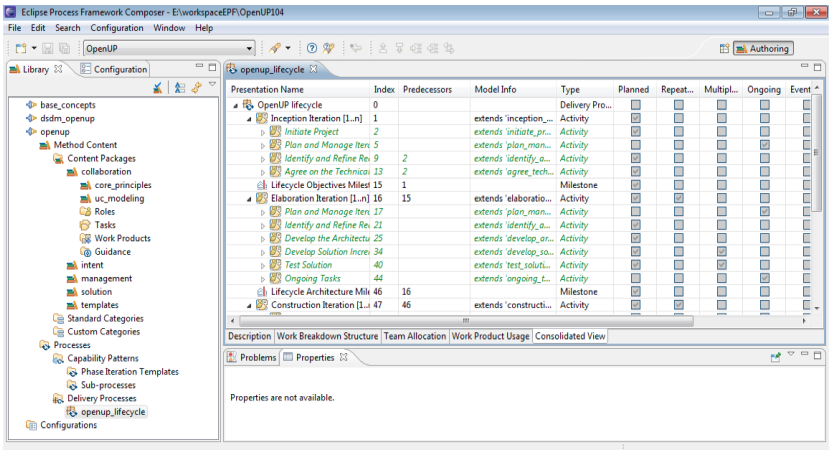

Figura 7.2: Librería de métodos de OpenUP en EPF

manual del modelo anterior para ajustarse más fielmente a las directrices de esta metodología y conseguir una estructura coherente de productos de trabajo. Entre otros ajustes, se han creado nuevas secciones y se han clasificado los artefactos como modelos o especificaciones (textuales o listas de elementos). El resultado de este proceso manual lo podemos comprobar en la figura 7.3.

Después de depurar el modelo de productos de trabajo, es momento de generar los modelos para las herramientas. Dado que el metamodelo de *Enterprise Architect* se compone únicamente del metamodelo de las herramientas de modelado visual (VMM), la derivación se llevó a cabo en un único paso. En la figura 7.4 podemos observar el modelo generado para *Enterprise Architect*, compuesto por paquetes, diagramas UML y los elementos de modelado correspondientes. Este modelo puede ser refinado, si se desea, antes de ejecutar el código necesario para generar una plantilla de proyecto (fichero con extensión *eap*), lista para ser usado en *Enterprise Architect*.

Al abrir el fichero generado con la herramienta (figura 7.5) podemos visualizar todos los elementos de trabajo necesarios para desarrollar la documentación de los proyectos software según *OpenUP*. En la figura 7.6 se muestra el entorno de trabajo de *Enterprise Architect* editando un diagrama de ejemplo UML.

Uno de los beneficios que ofrece MDE es la posibilidad de generar múltiples modelos específicos de plataforma desde un mismo modelo de entrada. Así pues, se invocó el motor de transformación ATL para derivar desde el modelo de productos de trabajo un modelo para *MediaWiki*. La derivación y la composición del modelo final se realizó en un único proceso de transformación. El modelo resultante de este proceso lo podemos visualizar en la figura 7.7.

Hay que tener en cuenta que, a diferencia de *Enterprise Architect*, *MediaWiki* es una herramienta genérica de edición colaborativa, por lo que no permite editar modelos, almacenando únicamente textos e imágenes en los artículos. Por ello, los modelos se materializan en imágenes estáticas. Sin embargo, esta solución podría mejorarse utilizando una *MediaWiki* con la extensión *GraphViz*[2] instalada para poder dibujar y editar los modelos, aunque esto ha quedado fuera del alcance de la tesis.

A partir del modelo generado en el paso anterior, ya se está en disposición de ejecutar los scripts necesarios para parametrizar la instalación de la wiki. En la figura 7.8 podemos observar un artículo de la wiki representando el documento de requisitos para un proyecto bajo *OpenUP* y en la figura 7.9 se muestra la categoría wiki para clasificar todos los productos de trabajo de la metodología.

En este caso de estudio, se ha ilustrado cómo se pueden generar automá-

[2] http://www.mediawiki.org/wiki/Extension:GraphViz

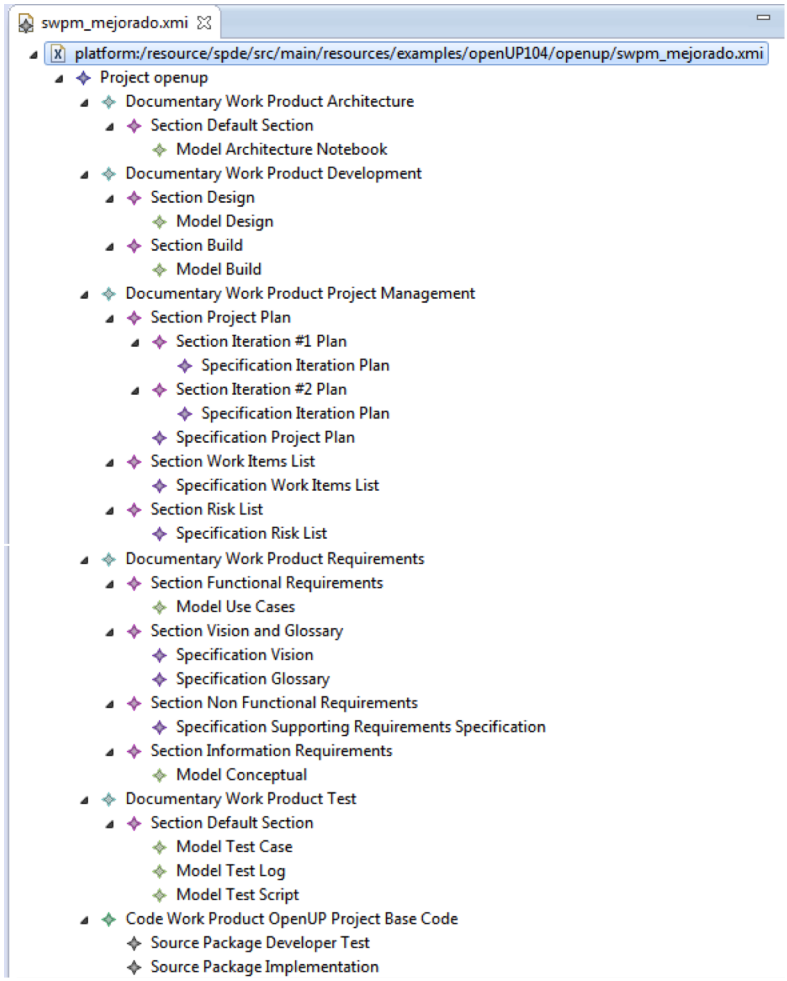

Figura 7.3: Modelo de productos de trabajo de OpenUP

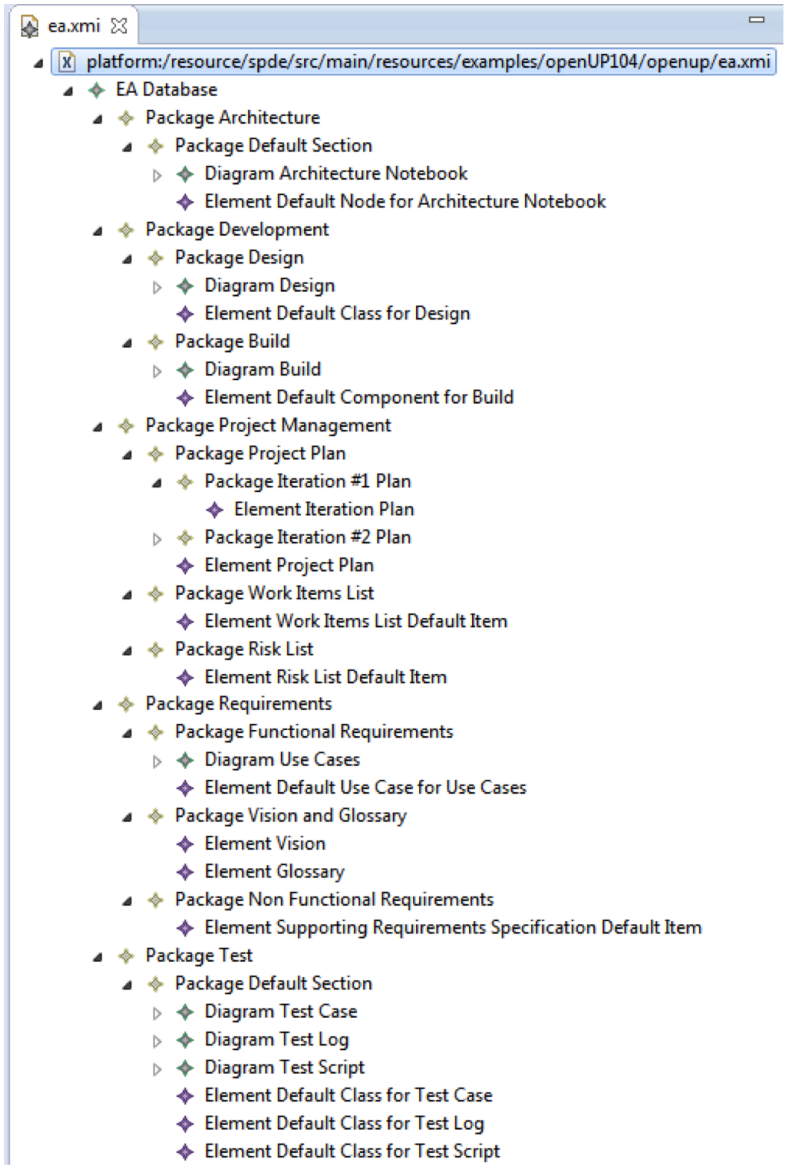

Figura 7.4: Modelo de Enterprise Architect para OpenUP

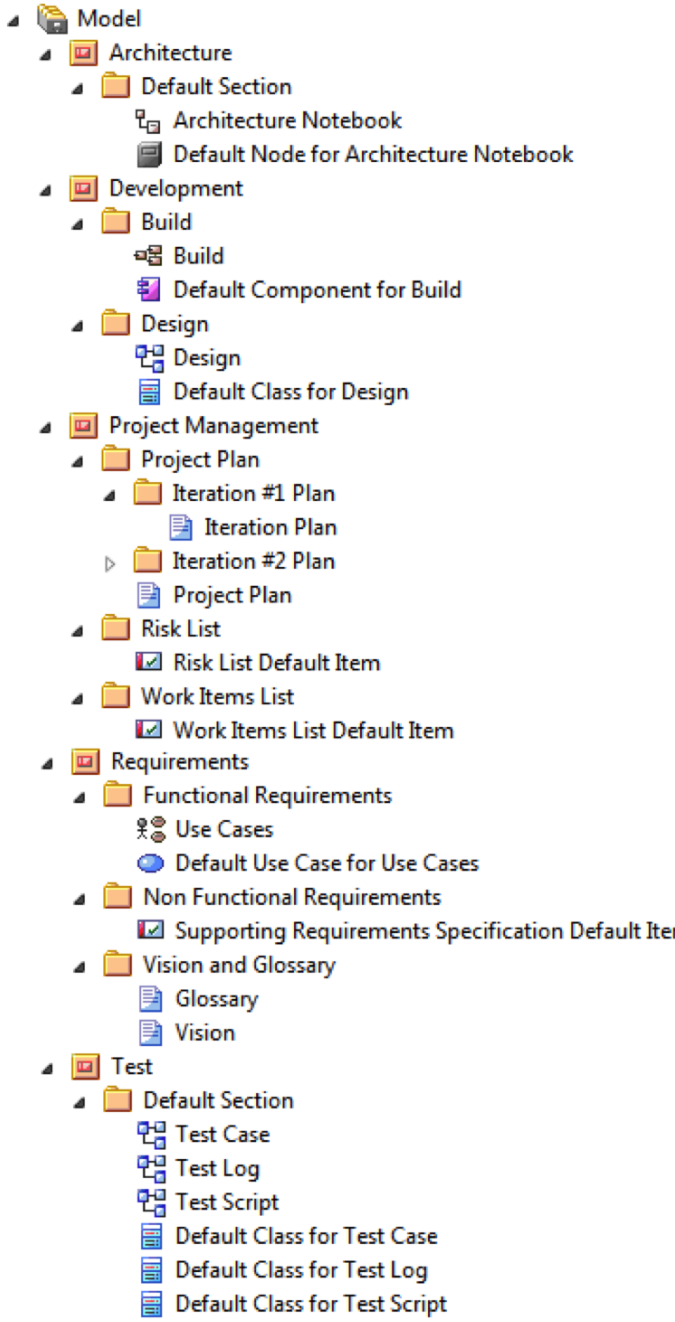

Figura 7.5: Proyecto Enterprise Architect para OpenUP

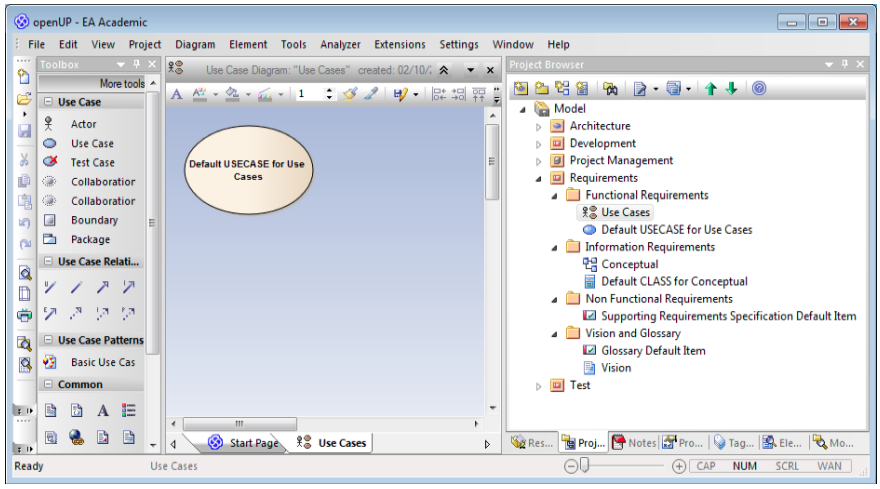

Figura 7.6: Entorno Enterprise Architect con la especificación de requisitos en OpenUP

ticamente plantillas para dos herramientas de soporte, *Enterprise Architect* y *MediaWiki*. De esta forma, se consigue normalizar la estructura de los productos de trabajo, según lo establecido en la metodología *OpenUP*.

7.2. Análisis de indicadores sobre habilidades de las personas

Con el fin de confirmar la viabilidad y la solidez del enfoque propuesto de apertura de datos y del desarrollo de soluciones de integración dirigidas a la evaluación de software, se presenta a continuación un caso de estudio relativo a la evaluación de indicadores sobre habilidades de las personas implicadas en proyectos software.

En los proyectos colaborativos de desarrollo de software es preciso monitorizar la actividad de los diferentes miembros de los equipos de trabajo, para así poder evaluar su desempeño. Habitualmente, los procedimientos de evaluación se basan en la inspección manual de la autoría y de la calidad de los entregables del proyecto. Sin embargo, dicha tarea resulta ser complicada especialmente

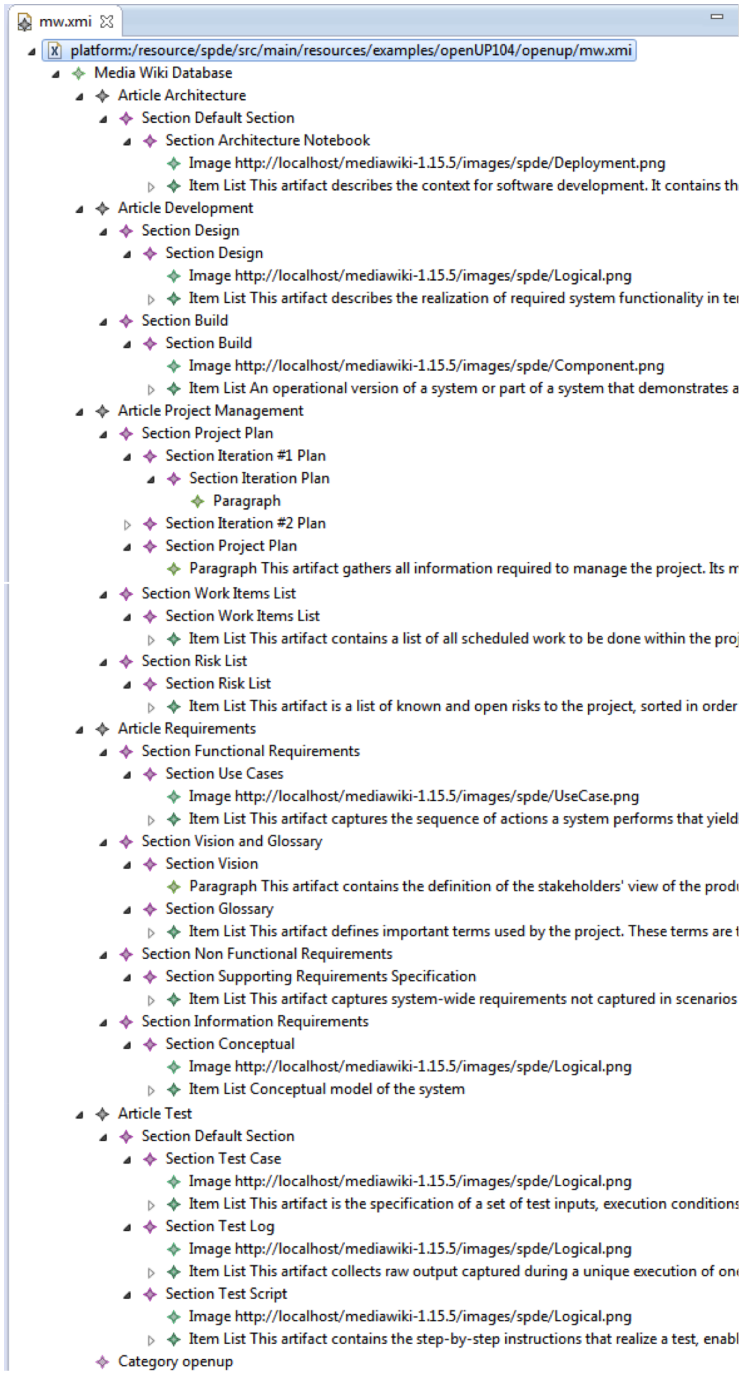

Figura 7.7: Modelo de MediaWiki para OpenUP

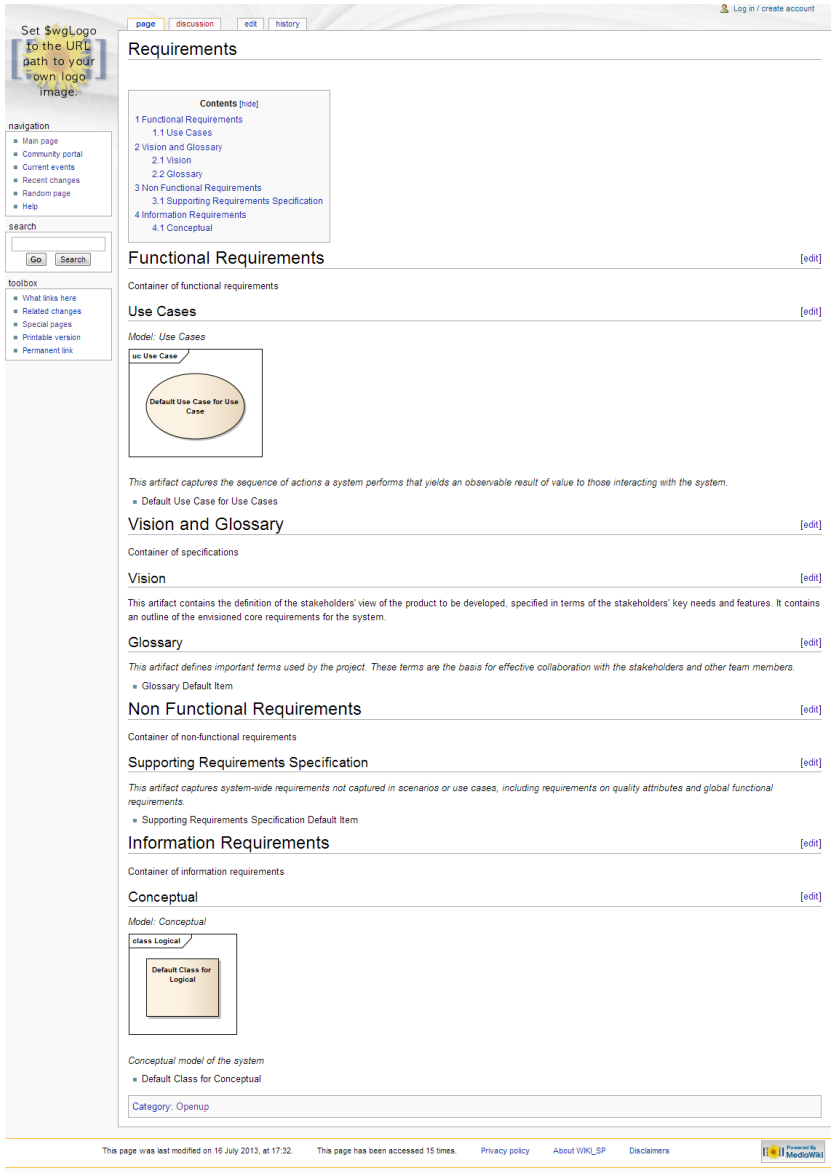

Figura 7.8: Artículo en MediaWiki con la especificación de requisitos en OpenUP

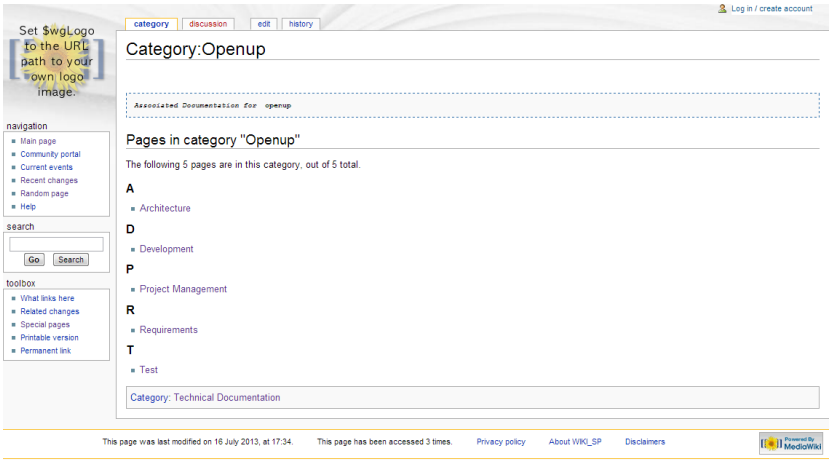

Figura 7.9: Categoría en MediaWiki con los documentos de OpenUP

cuando el número de equipos de trabajo y/o de proyectos es elevado. Por este motivo, se necesita disponer de una estrategia de evaluación automatizable y sostenible, así como unos criterios de evaluación adecuados.

Habitualmente, las métricas clásicas de software relativas a recursos de personal son la productividad y la experiencia. Sin embargo, existen otras métricas relacionadas que miden el desarrollo de ciertas competencias [47], como por ejemplo: equilibrio de la carga de trabajo, conocimiento de las herramientas, cumplimiento de hitos, liderazgo, adaptabilidad en la planificación y resolución de problemas, entre otras. El comportamiento de los usuarios a la hora de utilizar las herramientas de soporte permite revelar algunas de las prácticas anteriores, ya que en estas herramientas se almacenan evidencias con respecto a tales prácticas.

7.2.1. Diseño de la solución de integración ETL

Este caso de estudio se llevó a cabo en el contexto de la asignatura Ingeniería Web, correspondiente a 5º curso de Ingeniería Informática en la Universidad de Cádiz. En esta asignatura, los alumnos organizados en grupos tenían que desarrollar colaborativamente una aplicación web, utilizando el sistema de control

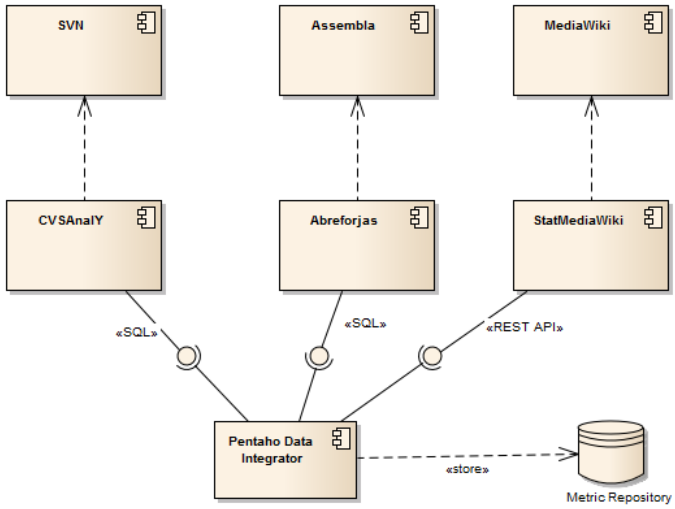

Figura 7.10: Solución ETL para la evaluación de métricas

de versiones SVN, el sistema de gestión de tareas *Assembla* y una instancia de *MediaWiki* para alojar los entregables documentales del proyecto.

A continuación, se describe la solución de integración de datos diseñada para obtener algunas de las métricas relativas a las habilidades personales. Esta solución (véase la figura 7.10), basada en el enfoque ETL, se implementó mediante la suite de inteligencia de negocios *Pentaho*, específicamente utilizando el entorno *Pentaho Data Integrator*[3]. El proceso diseñado consta de los siguientes pasos:

1. En primer lugar, se realiza una descarga de todos los ficheros de código fuente de los proyectos a analizar y se ejecuta el proceso de *CVSAnalY*, una herramienta dedicada a extraer información de los *logs* asociados a los ficheros bajo control de versiones.

2. Se lanza el proceso de análisis de *StatMediaWiki*, una herramienta para la extracción de métricas sobre la actividad de una determinada instancia de *MediaWiki*.

[3] http://kettle.pentaho.com/

3. Posteriormente, haciendo uso de la herramienta *Abreforjas*, descrita en el capítulo anterior, se extraen, normalizan y se obtienen métricas de utilización del sistema de gestión de tareas de *Assembla*.

4. Todas las métricas recopiladas anteriormente son transformadas en un modelo común y almacenadas en una base de datos de métricas.

5. Se ejecutan las operaciones estadísticas necesarias para calcular los indicadores definidos.

7.2.2. Obtención de métricas

A continuación, se describen los indicadores tratados en este caso de estudio y su procedimiento de cálculo. Una descripción ampliada del presente caso de estudio se encuentra en [182].

Asignación equilibrada de tareas a los miembros del equipo de proyecto

Este indicador permite comprobar habilidades relativas a la organización y planificación del trabajo en equipo, especialmente en metodologías ágiles donde se promueve el desarrollo de equipos autogestionados. Este indicador se calcula a partir del número de tareas asignadas a cada miembro del equipo y la cantidad de texto que ha introducido en ediciones sobre artículos de la wiki. El número de tareas asignadas también sería útil como indicador de la capacidad para la resolución de problemas, en el caso que se considerasen únicamente aquellas tareas clasificadas como error o incidencia.

En la tabla 7.2 se muestra la distribución del número de tareas asignadas a los miembros de un determinado proyecto y en 7.3 las contribuciones de cada uno de ellos a la wiki asociada [137]. A la vista de los resultados, se puede llegar a la conclusión de que el reparto de trabajo no ha sido equilibrado, como demuestra la alta desviación típica existente en las tareas y ediciones wiki de los miembros del equipo de proyecto.

Utilización de las herramientas de soporte

Este indicador permite comprobar el conocimiento que los usuarios tienen de las herramientas del ecosistema de soporte para el desarrollo del software. El indicador se calcula en base al número de *issues* registrados en la herramienta de gestión de tareas y al número de *commits* registrados en el sistema de control

Tabla 7.2: Ejemplo de asignación de tareas a miembros del equipo

Miembro	Tareas asignadas
Member1	9
Member2	4
Member3	9
Member4	27

Tabla 7.3: Ejemplo de contribuciones de los usuarios a la wiki

Miembro	Ediciones	Tasa de ediciones	Bytes	Tasa de bytes
Member1	101	51,5 %	44.998	63,6 %
Member2	59	30,1 %	13.945	19,7 %
Member3	21	10,7 %	4.953	7,0 %
Member4	15	7,7 %	6.875	9,7 %

de versiones. Sin embargo, un número elevado de *issues* o *commits* no implica una correcta utilización de la herramienta. Su efectividad depende de que su uso sea continuo en el tiempo y sin grandes periodos de inactividad.

En la tabla 7.4 se muestra el ritmo de finalización de tareas y en la tabla 7.5 se aprecia la diferencia de tiempo entre *commits* consecutivos de un proyecto en particular. Los periodos de inactividad mostrados en dichas tablas, permiten conjeturar la aparición de algún tipo de problema en el desarrollo del proyecto durante fechas determinadas.

Cumplimiento con las fechas de las tareas y de los hitos del proyecto

Este indicador permite comprobar el cumplimiento de los compromisos por parte de los miembros del proyecto, así como la organización y planificación del equipo.

Tabla 7.4: Ejemplo de actividad en sistema de gestión de tareas

Tarea	Fecha de cierre	Días transcurridos
1	15/01/2012	-
2	24/01/2012	9
3	02/02/2012	9
4	18/02/2012	16

Tabla 7.5: Ejemplo de actividad en control de versiones

Commit	Fecha	Inactividad
1	14/01/2012	-
2	14/01/2012	0
3	15/01/2012	1
4	17/01/2012	2
5	17/01/2012	0
6	17/01/2012	0
7	22/01/2012	5

Tabla 7.6: Ejemplo de retrasos en el cumplimiento de los hitos

Hito	Fecha de fin estimada	Fecha de fin real	Retraso
Sprint 0	18/04/2012	20/04/2012	2
Sprint 1	02/05/2012	02/05/2012	0
Sprint 2	09/05/2012	22/05/2012	13
Sprint 3	16/05/2012	22/05/2012	6
Sprint 4	23/05/2012	11/06/2012	19
Sprint 5	30/05/2012	03/06/2012	4
Sprint 6	13/06/2012	04/06/2012	-9

Este indicador puede obtenerse midiendo el retraso medio en la finalización de la última tarea asignada a cada hito del proyecto. En la tabla 7.6 podemos ver algunos datos referentes a un cierto proyecto.

7.3. Automatización de revisiones de calidad

Esta sección presenta un escenario detallado de utilización del *framework*, con el objetivo de ilustrar los pasos necesarios para automatizar la realización de revisiones técnicas durante el desarrollo de los proyectos software. Para ello, se necesita una solución de integración que, haciendo uso de los datos RDF expuestos por cada una de las herramientas de soporte, permita lanzar consultas SPARQL para automatizar la recogida y evaluación de evidencias sobre el uso de las técnicas, herramientas y métodos de Ingeniería del Software. Estas herramientas deben ser previamente dotadas con algunos de los mecanismos de exposición de datos citados en el capítulo anterior y configuradas con los vocabularios adecuados.

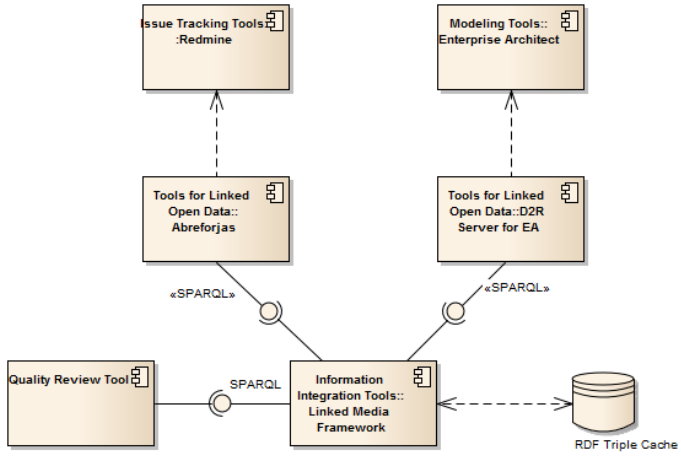

Figura 7.11: Solución EII para la automatización de revisiones técnicas

7.3.1. Diseño de la solución de integración EII

En las soluciones clásicas de integración EII se requiere un esquema virtual de datos que estructure de forma uniforme la información distribuida por las diferentes fuentes de datos. Sin embargo, en un enfoque LOD no se necesita disponer de un único esquema de datos, ya que los sistemas pueden publicar los datos simultáneamente con esquemas distintos. Asimismo, el lenguaje SPARQL permite realizar consultas sobre datos RDF conformes a múltiples vocabularios.

En la figura 7.11 se muestra la solución de integración de datos diseñada para este ejemplo. Esta solución, basada en el patrón de integración EII, se implementó haciendo uso de la plataforma *Linked Media Framework* (LMF), que incluye capacidades de almacenamiento, caché, versionado, razonamiento, indexación y consulta de datos, entre otras.

LMF dispone de un repositorio local de tripletas en el cual se cargaron los vocabularios incluidos en el *framework*. Posteriormente, se registraron en el razonador integrado en la plataforma las reglas de inferencia aplicables a los vocabularios. También se incluyeron las reglas típicas de razonamiento sobre los axiomas de equivalencia y especialización de *RDF Schema* y OWL.

A partir de la versión 1.1 de la especificación del lenguaje SPARQL se incluyó la característica de federación de consultas distribuidas. Gracias a esto, es posible dividir una consulta sobre diferentes fuentes de datos y posteriormente fusionar los resultados obtenidos. En nuestra solución de integración no hemos hecho uso de esta facilidad, ya que la propia plataforma LMF ofrece, además del repositorio local de datos RDF, un módulo para la recuperación y almacenamiento en caché, con el cual es posible cargar recursos RDF bajo demanda y de forma transparente, desde una serie de *datasets* registrados con anterioridad. Así pues, se registraron los *endpoints* SPARQL desde los cuales extraer los datos, en nuestro caso, los correspondientes al *dataset* habilitado para los proyectos de *Enterprise Architect* y al de la herramienta *Abreforjas*.

Posteriormente, se hace necesario establecer una vinculación entre cada uno de los identificadores (URI) que tienen los proyectos en las distintas herramientas de soporte. En el listado 7.1 se recoge un registro global de proyectos de una determinada organización ficticia, implementado como un conjunto de tripletas RDF.

```
@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix doap:
  <http://usefulinc.com/ns/doap#> .
@prefix dc:
  <http://purl.org/dc/elements/1.1/> .
@prefix owl:
  <http://www.w3.org/2002/07/owl#> .

<http://integration.my.org/resource/projects/foobar>
  rdf:type doap:Project ;
  dc:name "JAVA Web App" ;
  owl:sameAs <http://ea.my.org/resource/projects/foo> ;
  owl:sameAs <http://abreforjas.my.org/resource/projects/bar> .

<http://integration.my.org/resource/projects/openupTemplate>
  rdf:type doap:Project ;
  dc:name "Template Project for OpenUp Methodology" ;
  owl:sameAs <http://ea.my.org/resource/projects/openUp> ;
  owl:sameAs <http://abreforjas.my.org/resource/projects/openUp> .
```

Listado 7.1: Implementación en RDF de un registro global de proyectos

En ese extracto de código se muestra cómo un determinado proyecto de ejemplo es enlazado, mediante una relación de equivalencia, con los correspondientes

proyectos publicados en los *datasets* de *Enterprise Architect* y de *Abreforjas*. También se registra el proyecto plantilla resultante de desplegar previamente la metodología *OpenUP* sobre las herramientas de soporte.

7.3.2. Revisiones técnicas en proyectos de software

A partir de la solución de integración descrita anteriormente, se podrán desarrollar nuevas aplicaciones que permitan acometer revisiones de calidad sobre los proyectos software. A continuación, se van a presentar una serie de consultas SPARQL para ilustrar cómo sería la operativa interna de estas aplicaciones para la automatización de las revisiones.

La aplicación comenzaría con la preparación de una consulta, similar a la del listado de código 7.2, para obtener los identificadores o URI de los proyectos software que se quieren analizar. Las cláusulas de filtrado de esta consulta dependerán de los criterios de búsqueda que se ofrezcan en la interfaz de usuario de la aplicación.

```
PREFIX doap:
  <http://usefulinc.com/ns/doap#>
SELECT DISTINCT ?projectId ?projectName
WHERE{
  ?projectId rdf:type doap:Project .
  ?projectId dc:name ?projectName .
  FILTER regex(?projectName, "JAVA")
  }
ORDER BY ?projectName
```

Listado 7.2: Consulta SPARQL para obtener los identificadores de los proyectos

Con el objetivo de comprobar la correcta aplicación de ciertas prácticas de Ingeniería del Software, como las técnicas de modelado UML o la gestión de proyectos, se lanzan una serie de consultas SPARQL. Por ejemplo, con la consulta del listado 7.3 se pueden obtener los actores del sistema, identificados durante el análisis del proyecto, que no están asociados a ningún caso de uso. Otro ejemplo lo tenemos en la consulta del listado 7.4 dirigida a conocer si existen tareas no resueltas que hayan sido planificadas para hitos del proyecto cuya fecha límite ya venció.

```
PREFIX vmm:
  <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#>
SELECT ?actorId ?actorName
WHERE{
  <http://integration.my.org/resource/projects/foobar>
    vmm:packages/vmm:embeddedPackages*/ vmm:elements* ?actorId .
  ?actorId vmm:type "Actor" .
  ?actorId vmm:name ?actorName .

  MINUS {
    ?connId vmm:type "UseCase" .
    ?actorId vmm:connectors ?connId
  }  .
  MINUS {
    ?connId vmm:type "UseCase" .
    ?connId vmm:target ?actorId
  } .
  MINUS {
    ?connId vmm:type "Association".
    ?cduId vmm:type "UseCase" .
    ?actorId vmm:connectors ?connId .
    ?connId vmm:target ?cduId
  }  .
  MINUS {
    ?connId vmm:type "Association" .
    ?cduId vmm:type "UseCase" .
    ?connId vmm:target ?actorId .
    ?cduId vmm:connectors ?connId
   }
}
ORDER BY ?actorName
```

Listado 7.3: Consulta SPARQL para obtener aquellos actores que no tengan asociado ningún caso de uso

Además de llevar a cabo las comprobaciones anteriores, es posible verificar la adherencia de los proyectos con respecto a los procedimientos implantados en la organización. Por ejemplo, resulta sencillo comprobar si se han desarrollado los productos de trabajo esperados para un proyecto o si se han diseñado ciertos tipos de modelos UML dentro de alguno de los documentos técnicos. En el listado 7.5 se muestra una consulta con la cual se verifica si se ha generado el catálogo de requisitos del software. Realizando consultas similares a ésta sería posible conocer si se han elaborado el resto de productos de trabajo del proyecto.

Dado que en nuestra solución de integración los datos de las plantillas de

proyecto son también expuestos desde las herramientas de soporte, el diseño de las reglas de validación puede simplificarse, como podemos observar en el listado 7.6. Con esta consulta se obtienen los nombres de los productos documentales de la plantilla base del proceso, que no aparecen en el conjunto de productos generados en el proyecto bajo análisis.

```
PREFIX itm: <http://spi-fm.uca.es/spdef/models/genericTools/itm/1.0#>
SELECT ?versionName ?versionDueDate ?issueName ?issueCompletedDate
WHERE{
  <http://integration.my.org/resource/projects/foobar>
    itm:versions ?versionId .
  ?versionId a itm:Version .
  ?versionId itm:name  ?versionName .
  ?versionId itm:dueDate ?versionDueDate.
  ?versionId itm:issues ?issueId .
  ?issueId itm:name  ?issueName .
  ?issueId itm:completedDate ?issueCompletedDate .
  FILTER (?issueCompletedDate > ?versionDueDate)
}
ORDER BY ?issueDueDate
```

Listado 7.4: Consulta SPARQL para comprobar si todas las tareas vinculadas a un hito del proyecto completado están cerradas

```
PREFIX swpm:
  <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#>
SELECT DISTINCT ?productId ?productName
WHERE{
  <http://integration.my.org/resource/projects/foobar>
    swpm:workproducts ?productId .
  ?productId swpm:name ?productName .
  FILTER regex(?productName, "Requisitos")
  }
```

Listado 7.5: Consulta SPARQL para comprobar si el documento de requisitos

```
PREFIX swpm: <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#>
SELECT ?productName
WHERE{
  <http://integration.my.org/resource/projects/openupTemplate>
    swpm:workproducts ?productId .
  ?productId swpm:name ?productName .
  MINUS {
    <http://integration.my.org/resource/projects/foobar>
      swpm:workproducts ?productId .
    ?productId swpm:name ?productName
  }
}
ORDER BY ?productName
```

Listado 7.6: Consulta SPARQL para comprobar qué productos de trabajo no se han elaborado, según lo especificado en el proceso

Con las consultas anteriores hemos ilustrado algunas de las posibilidades que ofrece el enfoque LOD para la evaluación de procesos software. La elección de unos vocabularios u otros a la hora de diseñar las consultas SPARQL dependerá del nivel de detalle deseado para las evidencias que se quieran recolectar. En los primeros ejemplos, se han utilizado los vocabularios de herramientas genéricas VMM e ITM para obtener evidencias sobre el uso de las técnicas de modelado UML y sobre el seguimiento de tareas del proyecto, respectivamente.

Para comprobar la existencia de los productos de trabajo se utilizó el vocabulario SWPM. La realización de consultas utilizando este vocabulario es especialmente recomendable en aquellas situaciones donde los productos de trabajo de los proyectos se almacenen tanto en herramientas de modelado visual como en sistemas wiki. De esta forma, independientemente del tipo de herramienta empleada, el acceso a los datos sería siempre uniforme.

Parte IV

Epílogo

Capítulo 8

Conclusiones

Después de detallar SPDEF, el marco de trabajo para el despliegue y evaluación de procesos software sobre herramientas de soporte, es momento de presentar las conclusiones y las líneas de trabajo futuras que se derivan de la presente investigación. Asimismo, se incluyen las contribuciones realizadas en este contexto.

8.1. Conclusiones

En los últimos años han surgido diversos lenguajes de descripción de procesos que pretenden mejorar la interpretación y comprensión de los mismos por parte de todos los implicados. Con el objetivo común de representar con exactitud los procesos de desarrollo o mantenimiento de software, aparecieron algunos lenguajes que comparten ciertas características comunes en Ingeniería del Software, como las actividades, recursos, productos de trabajo, actores, herramientas y reglas asociadas a los procesos [18]. De esos lenguajes, SPEM, la propuesta de la OMG, es el más popular y extendido para la representación de procesos software.

El primer objetivo (*OBJ1*) planteado en esta tesis doctoral es el de conocer el grado de aceptación del metamodelo SPEM y explorar sus usos y aplicaciones mediante un exhaustivo estudio de la literatura. De los 373 artículos localizados en los motores de búsquedas más importantes, sólo fueron 115 los considerados como estudios primarios y que por tanto fueron analizados. Prácticamente la mitad de los trabajos encontrados fueron clasificados como *Off-topic*, dado que sólo utilizaban SPEM como mera notación para representar ciertas actividades

en el propio contexto de la investigación descrita en cada publicación. En los trabajos analizados se encontraron descripciones de procesos, nuevas técnicas, herramientas, metamodelos, transformaciones y *mappings* entre modelos. Estos estudios fueron clasificados según el ámbito de la gestión de procesos software a la que contribuyen: modelado, adaptabilidad (*tailoring*), verificación y validación, configuración y despliegue para la ejecución (*enactment*) y evaluación.

Después de un riguroso proceso de extracción y clasificación de datos, se pudieron confirmar algunos hechos interesantes. Por un lado, hay que destacar la amplia aceptación del metamodelo SPEM para el desarrollo de procesos y metodologías en áreas diversas, especialmente en metodologías orientadas a agentes, líneas de productos software, desarrollo ágil de software, UP y sistemas en tiempo real. Por otra parte, destacar también el importante volumen de trabajos que proponen extensiones a SPEM para cubrir ciertas carencias del lenguaje. Entre otras, se han propuesto extensiones para modelar aspectos de algunas metodologías concretas, ofrecer soporte para la construcción de líneas de procesos software y mejorar la ejecutabilidad del propio lenguaje.

Existen pocas contribuciones relativas a la configuración y la evaluación de procesos software, por lo que se hizo evidente la necesidad de afrontar mayores esfuerzos de investigación en estos ámbitos. La falta de alineamiento entre las definiciones de procesos y las herramientas de soporte, y la complejidad para la evaluación automatizada de procesos software son los problemas que han motivado el desarrollo de la presente investigación.

Los objetivos *OBJ2* y *OBJ3* de esta tesis se han materializado en un marco de trabajo para el despliegue y evaluación de procesos software. Este *framework* se basa en la aplicación de los principios y tecnologías de la Ingeniería del Software dirigida por modelos (MDE) y del paradigma de datos abiertos enlazados (LOD). De esta forma, se pretende cubrir el despliegue automatizado de los procesos software sobre herramientas de soporte y su posterior evaluación a partir de la integración de las métricas y evidencias publicadas por dichas herramientas.

En la figura 8.1 podemos ver una imagen ilustrativa del marco de trabajo propuesto. En ella, se representa cómo utilizando tecnologías MDE como ATL, se transforman modelos SPEM en adaptaciones particulares para diversas herramientas de soporte, como *Redmine*, *MediaWiki* o *Enterprise Architect*. Al mismo tiempo, estas herramientas expondrán sus datos en RDF de forma que puedan integrarse con otros sistemas, a través de vocabularios compartidos LOD. Gracias a esto, se podrán crear terceras aplicaciones o *mashups* para la evaluación de procesos software, mediante la explotación de los datos utilizando el lenguaje de consulta SPARQL.

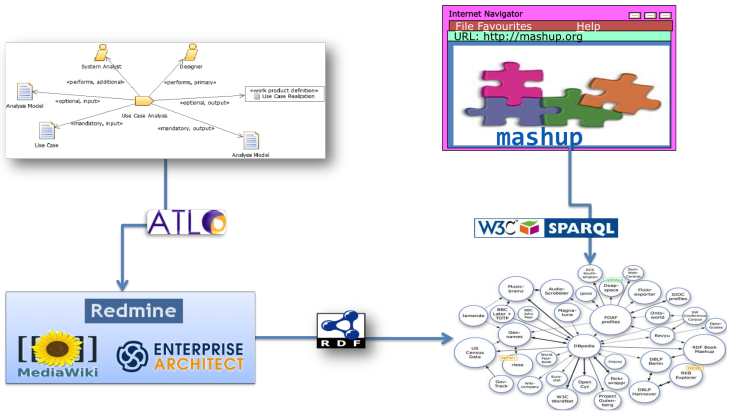

Figura 8.1: Framework para el despliegue y evaluación de procesos

Además del método para el despliegue y evaluación de procesos, el *framework* incluye una serie de modelos, implementados como metamodelos *Ecore* y como vocabularios *RDF Schema*; relaciones entre los modelos, implementadas como transformaciones ATL, axiomas RDF de equivalencia o especialización y reglas de inferencia KiWi; y varios componentes software para el despliegue automatizado y para la apertura de datos.

A continuación se describe cómo han sido validadas las hipótesis planteadas al comienzo de la investigación.

Hipótesis 1
Las inconsistencias entre la definición de los procesos y la ejecución de los proyectos podrían minimizarse, en parte, mediante la personalización y adaptación de las herramientas de soporte y la creación de plantillas específicas para las mismas.

La actividad *Adaptación de las herramientas de soporte* del método descrito en esta tesis tiene por objetivo automatizar la adaptación de las herramientas. Con el objetivo de aportar validez a la hipótesis anterior, el *framework* incluye una serie de componentes software para adaptar de manera semi-automática las herramientas *MediaWiki* y *Enterprise Architect* conforme a los productos de trabajo identificados en una determinada metodología. Asimismo, se ha descrito

un caso de estudio relativo al despliegue de la metodología *OpenUP* sobre las herramientas anteriores.

Hipótesis 2
Conseguir una visión global y uniforme de la información gestionada por las herramientas de soporte permitiría automatizar la recogida de métricas y la evaluación de la calidad en los procesos software.

La actividad *Apertura de las herramientas de soporte* del método descrito en esta tesis pretende ampliar las capacidades de las herramientas de soporte para que puedan exponer los datos que gestionan en un formato accesible y apto para la explotación posterior por parte de terceras aplicaciones. Para ello, se han implementado componentes software para la apertura de datos RDF desde herramientas de gestión de tareas, aplicaciones web basadas en MVC y para la herramienta de modelado *Enterprise Architect*. Una vez las herramientas de soporte están preparadas para la apertura de datos, se puede llevar a cabo la cuarta actividad propuesta en el método: *Desarrollo de Soluciones de evaluación*. Para validar esta hipótesis, se ha llevado a cabo un caso de estudio consistente a la evaluación de métricas sobre habilidades de las personas implicadas en los proyectos de software y se ha descrito en detalle un escenario descriptivo de integración de datos para la automatización de revisiones de calidad sobre los proyectos.

8.2. Líneas de trabajo futuras

En esta tesis se propone un *framework* para el despliegue y evaluación de procesos software. Sin embargo, no se trata de un *framework* cerrado, sino que ofrece la posibilidad de ir extendiéndolo, como se describe a continuación.

8.2.1. Incorporación de nuevos modelos

Este *framework* incluye dos modelos de nivel PIM: uno para la definición de productos de trabajo, SWPM, y otro para el control de proyectos, SPCM. Ambos modelos permiten perfilar elementos de los modelos de procesos software SPEM para su despliegue sobre herramientas de soporte. Sin embargo, existen otros aspectos relacionados con el proceso software que son vagamente tratados en el estándar SPEM, como por ejemplo, la gestión de la configuración o la gestión de personas.

Con respecto al primer punto, el establecimiento de los mecanismos de gestión de la configuración es esencial en los proyectos de desarrollo o mantenimiento de software. Estos aspectos suelen ser gestionados desde sistemas de integración continua, repositorios de componentes software y mediante sistemas de control de versiones.

Con respecto a la gestión de personas, en el capítulo anterior se presentó un caso de estudio relativo a la evaluación de ciertos indicadores relativos a las habilidades de los participantes en proyectos de desarrollo. Sin embargo, existen otros muchos aspectos a tener en cuenta, como por ejemplo, la estructura de la propia organización, los roles que desempeñan las personas, la gestión del conocimiento, la gestión de las habilidades o competencias, etc. Sobre esto último, podemos destacar el *e-Competence framework* [33] de la Comisión Europea, una clasificación de competencias relacionadas con las tecnologías de la información. Los sistemas de planificación de recursos humanos y las plataformas de aprendizaje online son ejemplos de herramientas que ofrecen soporte a la gestión de personas.

Así pues, la gestión de la configuración y la gestión de personas han quedado fuera del alcance de la presente tesis, por lo que el diseño de los correspondientes modelos de nivel PIM y PSM, y el posterior despliegue sobre herramientas específicas, se plantean como líneas de trabajo futuras.

8.2.2. Desarrollo de nuevas herramientas

Hasta el momento, el componente SPDT para el despliegue de procesos solo opera con las herramientas *MediaWiki* y *Enterprise Architect*; el despliegue sobre la herramienta *Redmine* se encuentra en desarrollo. Asimismo, se pretende realizar el despliegue de procesos sobre wikis enriquecidas, como *Semantic MediaWiki*[1], una extensión de *MediaWiki* con capacidades adicionales como la inclusión de anotaciones semánticas, formularios de introducción de datos y exportación en formato RDF, entre otras.

Adicionalmente, se está trabajando en un software para coordinar las actividades de transformación y composición de modelos necesarias para facilitar el despliegue de los procesos sobre herramientas concretas. Este software permitiría asistir al usuario en estas labores, mediante una interfaz visual similar a la de la figura 8.2 y siguiendo la idea de *GMF Dashboard*[2], el *plugin* de *Eclipse* dedicado a controlar el flujo de trabajo necesario para la construcción de DSLs visuales.

[1] http://semantic-mediawiki.org/
[2] http://eclipse.org/gmf-tooling/

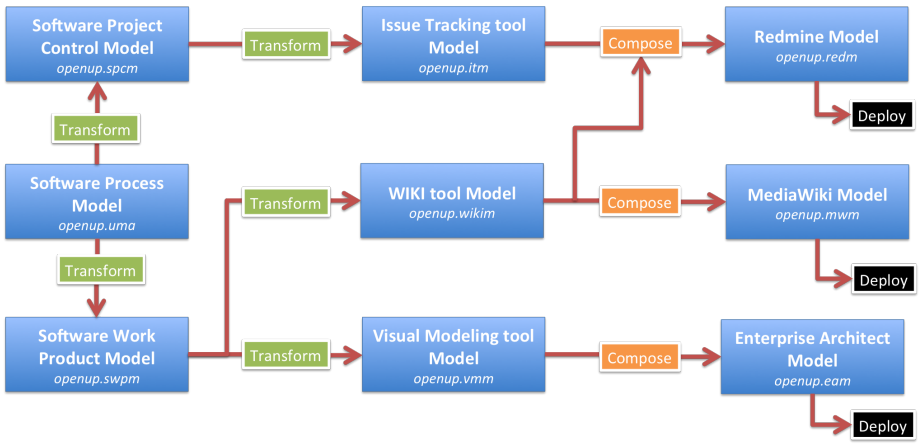

Figura 8.2: Flujo de trabajo para el despliegue de procesos

Además de lo anterior, se está desarrollando un software para la automatización de revisiones de calidad, mediante la obtención de métricas y evidencias desde las herramientas de soporte, como se propone con la actividad *Desarrollo de soluciones de evaluación* del método propuesto en la tesis.

8.2.3. Experimentación en Ingeniería del Software

Las organizaciones TIC con un alto nivel de madurez en sus procesos productivos suelen dedicar esfuerzos y recursos para la medición y análisis de sus procesos. Habitualmente, estas organizaciones implementan iniciativas para la definición, implantación, evaluación y mejora de la calidad de los procesos, como ISO/IEC 12207 o CMMI. Por contra, los resultados que obtienen no suelen difundirse públicamente y por tanto, sólo son utilizados para la mejora interna de esas organizaciones. Sólo aquellas que promueven el conocimiento abierto y que apuestan por la investigación en Ingeniería del Software divulgan sus resultados. Por ello, resulta complicado realizar un análisis global sobre los beneficios reales que se pueden obtener aplicando determinadas técnicas, herramientas, lenguajes o métodos de Ingeniería del Software.

Hace unos años, la Universidad de Ottawa inició una interesante iniciativa

denominada *PROMISE Software Engineering Repository* [172] dirigida a la recopilación de datos de proyectos software. Su objetivo fundamental es dotar a los investigadores de un volumen de datos suficiente para la creación de modelos predictivos de software que sean verificables, repetibles, refutables y mejorables. Sin embargo, a pesar de las ventajas que pueda suponer disponer de un repositorio como éste, *PROMISE* no llegó a alcanzar una gran aceptación en la comunidad de Ingeniería del Software.

Por otra parte, las forjas de software almacenan gran cantidad de información sobre proyectos. Por ejemplo, la plataforma *SourceForge* almacena datos de más de 300.000 proyectos hasta la fecha. Toda esa información puede utilizarse como base empírica de datos para la experimentación en Ingeniería del Software, como han hecho algunos autores [77, 151]. No obstante, la mayor parte de la información recogida en las forjas es a menudo incompleta debido a la propia naturaleza distribuida y descentralizada de muchos proyectos de software libre, a la poca aplicación de estándares de proceso durante el desarrollo y al escaso detenimiento a la hora de enriquecer con meta-información los datos de los proyectos. Por ejemplo, en el caso de las tareas o *issues*, éstas suelen incluir una descripción, un estado, la persona que la realiza, los tiempos estimados y dedicados, etc., pero no es común que estén categorizadas de forma adecuada, por lo cual, es complicado discernir si se trata de una tarea de análisis de requisitos, si es el diseño o la implementación de un determinado módulo del software, si se trata del desarrollo de una prueba o de cualquier otra actividad.

Como trabajo futuro, se espera aplicar y extender el *framework* propuesto con vistas a la experimentación en Ingeniería del Software. Para ello, se pretenden mejorar los mecanismos para la apertura de datos RDF desde las herramientas de soporte, mediante la aplicación de técnicas de reconocimiento de lenguaje natural, y mejorar los procedimientos de evaluación, mediante la utilización de cubos *On-Line Analytical Processing* (OLAP) y de técnicas de minería de datos.

8.3. Contribuciones

La realización de las actividades de la tesis doctoral han estado contextualizadas en diversos proyectos de investigación correspondientes a convocatorias nacionales e internacionales de financiación pública. Son los siguientes:

- *Asceta: Accesibilidad a servicios semánticos para la formación a través de contenidos educativos y tecnologías del aprendizaje.* Proyecto Excelencia de la Junta Andalucía (P09-TIC-5230).

- *Bestmark: Plataforma para el modelado, personalización y benchmarking en la mejora de procesos normalizados*. Proyecto Avanza 2 del MITYC (TSI-020100-2011-396).
- *eCultura: Desarrollo de una plataforma semántica para la explotación de contenido cultural*. Proyecto Avanza I+D del MITYC (TSI-020501-2008-53).
- *ODS: Open Discovery Space: A collaborative and multilingual open learning infrastructure designed to boost demand for Europe-wide eLearning Resources*. Proyecto de la Comisión Europea (CIP-ICT-PSP-2011-5).
- *VOA3R: Virtual Open Access Agriculture & Aquaculture Repository: Sharing Scientific and Scholarly Research related to Agriculture, Food, and Environment*. Proyecto de la Comisión Europea (ICT-PSP-250525).

A continuación se recogen las contribuciones desarrolladas con la presente investigación: artículos en revistas indexadas, publicaciones no indexadas y otras aportaciones originales.

8.3.1. Publicaciones indexadas de impacto

En esta sección se describen los principales artículos publicados en revistas clasificadas en los índices *Journal Citation Reports* (JCR) o *SCImago Journal Rank* (SJR), en el contexto de la investigación planteada en esta tesis.

Uses and applications of Software & Systems Process Engineering Meta-Model process models. A systematic mapping study

Datos de la revista:
International Journal of Software: Evolution and Process
Editorial: John Wiley & Sons, Ltd.
Factor de Impacto ISI JCR: 1.273
Ranking 2012 en JCR: 30/105 1er Tercil (Computer Science, Software Engineering)

Contexto de la publicación:
Este artículo muestra el diseño y los resultados de la revisión de la literatura realizada acerca de los usos y aplicaciones del metamodelo SPEM. La mayor parte del capítulo 3 de esta tesis se encuentra publicado en este artículo.

Cita:
[160] Ruiz-Rube, I., Dodero, J. M., Palomo-Duarte, M., Ruiz, M. and Gawn, D. (2013), Uses and applications of Software & Systems Process Engineering Meta-Model process models. A systematic mapping study. J. Softw. Evol. and Proc..

Palabras clave:
SPEM; software process engineering; systematic mapping study; model-driven engineering; business process management

Resumen:
Software process engineering is a discipline, which aims to study and improve software development and maintenance processes. The explicit definition of software processes is essential. To this end, the Object Management Group consortium proposed the Software & Systems Process Engineering Meta-Model (SPEM) that exploits the benefits of the Model Driven Architecture paradigm applied to software process models, instead of software specification models. The aim of this study is to discover evidence clusters and evidence deserts in the use and application of SPEM from a business process management point of view. To reach the proposed objective, we have undertaken a systematic mapping study of the existing scientific literature. The reviewed literature deals mainly with process modeling and, to a lesser extent, with process adaptability, verification, and validation, enactment and evaluation. Wide agreement exists in using the SPEM meta-model to develop different types of methods and processes. Further research efforts are needed in areas related to enactment and evaluation of software processes. There is a need to evolve to a new version of the meta-model that incorporates the improvements proposed by different authors.

Model-Driven Learning Design

Datos de la revista:
International Journal of Research and Practice in Information Technology
Editorial: Australian Computer Society
Factor de Impacto ISI JCR: 0.222
Ranking 2012 en JCR: 99/105 3er Tercil (Computer Science, Software Engineering)

Contexto de la publicación:

Este artículo presenta un método de desarrollo basado en modelos para el diseño de procesos de aprendizaje. Su aplicación, aunque diferente de la propuesta en esta tesis, se basa en los mismos principios descritos en la sección 6.1.2.

Cita:
[48] Dodero, J. M., Ruiz-Rube, I., Palomo-Duarte, M., & Cabot, J. (2012). Model-driven learning design. Journal of Research and Practice in Information Technology, 44(3), 267-288.

Palabras clave:
Learning design; model-driven development; domain-specific languages

Resumen:
Learning Design is a framework of elements that are used for the formal specification of learning courses. Learning Design languages have been defined to facilitate the editing of online courses, usually including a number of technical formalisms that have proved to be scarcely comprehensible to non-technical staff. In this work we describe a model-driven development approach based on a Learning Design domain-specific language that makes it possible to author and generate learning design courses and to export them into languages and formats of common learning environments and course players.

Connecting Closed World Research Information Systems through the Linked Open Data Web

Datos de la revista:
International Journal of Software Engineering and Knowledge Engineering
Editorial: World Scientific Publishing
Factor de Impacto ISI JCR: 0.295
Ranking 2012 en JCR: 96/105 3er Tercil (Computer Science, Software Engineering)

Contexto de la publicación:
En este artículo se describe cómo se pueden integrar los sistemas de información científica mediante LOD, haciendo uso de algunas de las recomendaciones para la publicación y consumo de datos presentadas en las secciones 6.1.3 y 6.1.4. En este caso, se utiliza la estrategia de implementar adaptadores de datos para la apertura de las herramientas.

Cita:
[85] Joerg, B., Ruiz-Rube, I., Sicilia, M. A., ... & Barriocanal, E. G. (2012). Connecting Closed World Research Information Systems through the Linked Open Data Web. International Journal of Software Engineering and Knowledge Engineering, 22(03), 345-364.

Palabras clave:
Current Research Information Systems; CRIS; Common European Research Information Format; CERIF; Linked Open Data; LOD; LD; Ontologies; Conceptual Modeling

Resumen:
Research Information Systems (RIS) play a critical role in the sharing of scientific information and provide researchers, professionals and decision makers with the required data for their activities. Existing RIS standards have proposed data models to represent the main entities for storage and exchange. These account for the needs of multiple stakeholders through a high flexibility based on a formal syntax and declared semantics, but for techno-historical reasons they assume the completeness of information within system boundaries. The distributed nature of research information across systems calls for a mechanism to link the local entities from the closed world of concrete RISs with other possibly underspecified entities exposed through other means, as for example, the Linked Open Data Web. By transformation of a relational model into an open graph model, differences between the two system paradigms are revealed. The main principles and techniques for exposing CERIF-driven relational data as linked data will be provided as a first step demonstrating effective RISs interconnection through the linked open data (LOD) Web.

Assessment of collaborative learning experiences by graphical analysis of wiki contributions

Datos de la revista:
Interactive Learning Environments
Editorial: Taylor & Francis Group
Factor de Impacto ISI JCR: 1.302
Ranking 2012 en JCR: 39/216 1er Tercil (Education, Educational Research)

Contexto de la publicación:
Este artículo describe un enfoque para la evaluación de las contribuciones que realizan los alumnos durante el desarrollo de proyectos de aprendizaje colaborativo asistido por wikis. El mecanismo de análisis de wikis presentado en este trabajo se aplica en la evaluación del desempeño de los miembros de los proyectos incluido en el caso de estudio de la sección 7.2.

Cita:
[137] Palomo-Duarte, M., Dodero, J. M., Medina-Bulo, I., Rodríguez-Posada, E. J., & Ruiz-Rube, I. (2012). Assessment of collaborative learning experiences by graphical analysis of wiki contributions. Interactive Learning Environments, 1-23

Palabras clave:
Computer-supported collaborative learning; Wikis; e-learning assessment; Data visualization; Graphical analysis tool

Resumen:
The widespread adoption of computers and Internet in our life has reached the classrooms, where computer-supported collaborative learning (CSCL) based on wikis offers new ways of collaboration and encourages student participation. When the number of contributions from students increases, traditional assessment procedures of e-learning settings suffer from scalability problems. In a wiki-based learning experience, automatic tools are required to support the assessment of such huge amounts of data. In this work, we present StatMediaWiki, a tool that collects and aggregates information that helps to analyze a MediaWiki installation. It generates charts, tables and different statistics enabling easy analysis of wiki evolution. We have used StatMediaWiki in a Higher Education course and present the results obtained in this case study.

Open linked data model revelation and access for analytical web science

Datos de la revista:
Communications in Computer and Information Science
Editorial: Springer Berlin Heidelberg
Índice de Impacto Scopus SJR: 0,140
Ranking 2012 en SJR: 145/192 3er Tercil (Computer Science, miscellaneous)

Contexto de la publicación:
Este trabajo detalla una de las estrategias para la apertura de herramientas de soporte presentadas en la sección 6.1.3, concretamente, la relativa a la implementación de extensiones para la publicación controlada de metadatos RDF, mediante la introspección automática del código fuente.

Cita:
[49] Dodero, J. M., Ruiz-Rube, I., Palomo-Duarte, M., & Vázquez-Murga, J. (2011). Open linked data model revelation and access for analytical web science. In Metadata and Semantic Research (pp. 105-116). Springer Berlin Heidelberg.

Palabras clave:
Linked data; Web science; Web application frameworks

Resumen:
The extension of regular web applications with linked data and the provision of specific web services to exploit their datasets is still a challenge. In this paper we describe a method to generate linked data and reveal them in a controlled manner for open source web applications. Revelation is carried out either at the controller or the model components of web applications that are based in the model-view-controller architecture. The approach facilitates the definition of access control policies and other non-functional requirements on the application objects and services, as well as the mapping of the application model to standard RDF schemata and vocabularies.

Accessing learning resources described in semantically enriched weblogs

Datos de la revista:
International Journal of Metadata, Semantics and Ontologies
Editorial: Inderscience Publishers
Índice de Impacto Scopus SJR: 0,424
Ranking 2012 en SJR: 56/146 2do Tercil (Social Sciences, Library and Information Sciences)

Contexto de la publicación:
En este trabajo se describe un escenario de integración de datos basado en tec-

nologías LOD para la recuperación de recursos de aprendizaje descritos en los contenidos alojados en un motor de blogs. El enfoque de integración de datos RDF utilizando el lenguaje de consulta SPARQL se basa en los mismos principios presentados en la sección 7.3.

Cita:
[158] Ruiz–Rube, I., Cornejo, C. M., & Dodero, J. M. (2011). Accessing learning resources described in semantically enriched weblogs. International Journal of Metadata, Semantics and Ontologies, 6(3), 175-184.

Palabras clave:
Linked data; Learning design, LMS integration; Learning Activity Management System; RDF

Resumen:
In this paper, we describe how to design learning activities that dynamically provide thematic Web resources from a linked-data repository. The aim is to enable teachers to share a variable set of resources related to a given subject and postpone the actual resource delivery to the deployment or enactment of the course. We have proposed a Learning Activity Management System (LAMS) tool that provides an interface to automatically select the related resources to be delivered to the students who are running a learning activity. The thematic resource repository was a linked-data extension of the WordPress blog engine. This extension allows enriching text-based and video information contained in blog entries with RDF triples that can be externally managed and exploited. Our approach allows discovering learning resources from external linked data sets and enrich blog contents with linked-data, independently of the underlying conceptual model.

8.3.2. Otras publicaciones

Además de los artículos anteriores, se han desarrollado otras contribuciones científicas. A continuación, se incluyen las referencias de las mismas.

Open data framework for sustainable assessment in software forges

Datos del congreso:
International Conference on Web Intelligence, Mining and Semantics, WIMS

2013
Editorial Actas: ACM New York

Contexto de la publicación:
En este trabajo se describe en profundidad el caso de estudio de la sección 7.2, incluyendo tanto el método de trabajo utilizado para la integración de datos, como el análisis de las métricas obtenidas.

Cita:
[182] Traverso-Ribón, I., Ruíz-Rube, I., Dodero, J. M., & Palomo-Duarte, M. (2013). Open data framework for sustainable assessment in software forges. In Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics (p. 20). ACM.

Palabras clave:
Project-based learning; Evaluation; software forge; ETL

Resumen:
In a project-based learning experience, the detailed monitoring of the activities in which team members participate can be useful to evaluate their work. Using learning-oriented assessment procedures, supervisors can assess the teamwork abilities with a formative purpose. Evaluation strategies such as self-assessment, peer assessment and co-assessment are often used to make evaluation formative and sustainable. Conducting an assessment strategy is not easy for team members, since they need before to have a reasonable understanding of the evaluation process and criteria. This paper describes a learning-oriented evaluation methodology and a open data framework that can be applied to collaborative software development project settings. An evaluation rubric and a series of indicators that provide evidences about the developed skills have been elaborated and applied in a small-scale project-based course on Web Engineering. Projects were managed and developed with the help of an open source software forge that contains a ticketing tool for planning and tracking of tasks, a version control repository to save the software deliverables, and using a wiki to host text deliverables. The experience provides evidences in favor of using the assessment method and open data framework to make teamwork evaluation more sustainable.

Non-functional aspects of information integration and research for the web science

Datos de la publicación:
International Conference on Computational Science, ICCS 2011
Editorial Actas: Inderscience Publishers

Contexto de la publicación:
En este trabajo se discuten aspectos no funcionales de la integración de información, como la veracidad o la evolución en el tiempo, a tener en cuenta a la hora de implementar soluciones de integración de datos procedentes de fuentes diversas (sección 6.1.4).

Cita:
[161] Ruiz-Rube, I., Dodero, J. M., & Stoitsis, J. (2011). Non-functional aspects of information integration and research for the web science. Procedia Computer Science, 4, 1631-1639.

Palabras clave:
Web science; Information integration; Semantic web; Linked data; Humanities; Ontologies

Resumen:
Web Science is emerging as an interdisciplinary field that views the Web as a relevant source of information to be analysed for diverse scientific purposes. Semantic and linked data techniques and standards have been used to integrate existing Web information developing e-research efforts. Web applications and sources have to publish enriched information and data models, which are then matched and aligned. Some aspects of this integration are related with the evolutionary tracking, trustiness and plurality of the information available for e-science research. These aspects need to be modeled independently of the information and data sources of the research discipline. Here we provide an integrative ontology that enables to model such non-functional, informational aspects, to integrate information sources from enriched linked data applications. Finally we describe an application case of information research in the e-culture field.

Evaluación de un ecosistema software en organizaciones de desarrollo web bajo CMMI

Datos de la publicación:
Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2010
Editorial Actas: Ibergarceta Publicaciones

Contexto de la publicación:
En este trabajo se presenta la evaluación de un conjunto de herramientas de soporte a la gestión y producción de software, citadas en la sección 4.1, para comprobar el soporte que ofrecen a los procesos software conformes a modelos de mejora como CMMI.

Cita:
[159] Ruiz-Rube, I., Cornejo-Crespo, C., Dodero, J. M., & Ruiz, M. (2010). Evaluación de un ecosistema software en organizaciones de desarrollo web bajo CMMI. In Actas de las Jornadas de Ingeniería del Software y Bases de Datos (pp. 237-248).

Palabras clave:
Ingeniería Web, CMMI, métodos ágiles, ecosistema software, calidad de software, herramientas.

Resumen:
Las organizaciones dedicadas al desarrollo y mantenimiento de aplicaciones web suelen emplear un enfoque ágil en la ejecución de sus proyectos. Si además, estas organizaciones quieren implantar un modelo para la mejora y evaluación de procesos software como CMMI-DEV sin perder la agilidad necesaria en este tipo de proyectos, deben al menos utilizar herramientas de soporte que permitan una gestión más eficiente de las buenas prácticas que proponen ambos modelos. Así pues, en este trabajo se describe un método para la evaluación de herramientas de soporte a CMMI en organizaciones dedicadas al desarrollo de aplicaciones web y además se detalla el estudio de un ecosistema tecnológico compuesto por varias herramientas de software libre con el fin de ilustrar la aplicación del método y evaluar la idoneidad de este ecosistema.

8.3.3. Aportaciones originales

De forma complementaria a las publicaciones científicas, se han elaborado un conjunto de aportaciones más específicas. Estas aportaciones, desarrolladas en el contexto del *framework* para el despliegue y evaluación de procesos software, se encuentran disponibles para su consulta y descarga en el sitio web de apoyo a la tesis[3]. Son las siguientes:

Diseño y resultados de la revisión de la literatura

Se ha desarrollado mediante el entorno EPF una biblioteca de métodos SPEM. Esta biblioteca ha sido publicada en formato HTML para su visualización y tiene por objetivo modelar los pasos necesarios para llevar a cabo un estudio de alcance de la literatura o SMS, en Ingeniería del Software. Asimismo, los resultados extendidos y completos del estudio de alcance realizado sobre los usos y aplicaciones de SPEM pueden encontrarse también en la web.

Método para el despliegue y evaluación de procesos software

El método para el despliegue y evaluación de procesos descrito en la sección 6.1 está modelado con los lenguajes SPEM y BPMN y se encuentra disponible para su descarga en formato XMI.

Implementaciones de los modelos y de sus relaciones

Todos los modelos descritos en el marco de trabajo descritos en la sección 6.2 están disponibles para su descarga en formato XMI, en forma de metamodelos *Ecore* y como vocabularios RDF. Asimismo, están disponibles las relaciones (sección 6.3) entre modelos implementadas como reglas de transformación en ATL, como axiomas RDF de equivalencia o especialización y como reglas de razonamiento para el motor KiWi.

Software desarrollado

Se proporciona acceso al software desarrollado para el despliegue asistido de procesos software (sección 6.4.1), incluyendo los *plugins* de *Eclipse* y los generadores de código para *MediaWiki* y *Enterprise Architect*. Del mismo modo, se proporciona acceso a los componentes para la apertura de datos descritos en la

[3] http://spi-fm.uca.es/spdef

sección 6.4.2: *Abreforjas*, *EasyData* y las configuraciones necesarias para *D2R Server*.

Casos de estudio

Se incluyen los modelos CIM, PIM y PSM descritos en la sección 7.1 que son necesarios para el despliegue de la metodología *OpenUP* sobre las herramientas de soporte. También se incluye la plantilla de proyecto generada automáticamente para *Enterprise Architect*, así como el acceso a una instancia de *MediaWiki* inicializada con las plantillas necesarias para trabajar con esta metodología.

El código fuente de los procesos de integración diseñados para la evaluación de métricas sobre habilidades de personal (sección 7.2) está disponible para su reutilización. Además, se incluyen en el sitio web los resultados del caso de estudio. Con respecto a la automatización de revisiones de calidad, se proporciona acceso al *endpoint* SPARQL sobre el cual se pueden lanzar las consultas descritas en la sección 7.3.

Parte V

Anexos

Anexo A

Resultados del estudio de alcance

Tabla A.1: Publicaciones sobre el modelado de procesos

Title	Research Type	Contribution Type	MetaModel Type	Modeling	Ref.
Experience-Based Approach for Adoption of Agile Practices in Software Development Projects	Experience	Process	SPEM 2.0 Extension	ASD	[100]
Agile Development of Web Application by Supporting Process Execution and Extended UML Model	Proposal	Framework	SPEM 1.1	ASD	[193]
Towards Tool Support for Situational Engineering of Agile Methodologies	Proposal	Process	SPEM 2.0	ASD	[1]
Development Process of Mobile Application SW Based on Agile Methodology	Proposal	Process	SPEM 2.0	ASD,MAD	[84]
XWebProcess: Agile Software Development for Web Applications	Proposal	Process	SPEM 1.1	ASD,WS	[164]
Method Library Framework for Safety Standard Compliant Process Tailoring	Validation	Process	SPEM 2.0	AS	[99]
Using a software process for ontology-based context-aware computing	Experience	Process	SPEM 1.1	CaS	[131]
Designing context-sensitive systems: An integrated approach	Proposal	Framework	SPEM 2.0	CaS	[188]
Odyssey-CCS: A Change Control System Tailored to Software Reuse	Proposal	Tool	SPEM 1.1	CM	[113]
A reusable traceability framework using patterns	Proposal	Framework	SPEM 1.1	CM	[90]
A software maintenance methodology for small organizations: Agile MANTEMA	Validation	Process	SPEM 2.0	CM	[146]
Requirements for a knowledge management framework to be used in software intensive organizations	Proposal	Framework	SPEM 1.1 Extension	CMMI	[120]
A Meta-Model for the CMM Software Process	Proposal	Model	SPEM 2.0 Extension	CMMI	[109]
The Tutelkan Reference Process: A Reusable Process Model for Enabling SPI in Small Settings	Proposal	Process	SPEM 1.1 Devised	CMMI	[70]
Applying UML and software simulation for process definition, verification, and validation	Proposal	Framework	SPEM 1.1 Devised	CMMI	[80]
Business Process Design and Implementation for Customer Segmentation e-Services	Proposal	Process	SPEM 1.1	CRM	[56]
A personalization process for spatial data warehouse development	Validation	Process	SPEM 2.0	DW	[69]
On the role of software process modeling in software ecosystem design	Philosophical	Model	SPEM 2.0 Extension	SOFTECO	[144]
Designing a Unified Process for Embedded Systems	Proposal	Framework	SPEM 1.1	ERT	[150]
Integration of PECOS into MARMOT for Embedded Real Time Software Component-Based Development	Proposal	Process	SPEM 1.1	ERT	[163]

Tabla A.1: Publicaciones sobre el modelado de procesos (cont.)

Title	Research Type	Contribution Type	MetaModel Type	Modeling	Ref.
A tool supported engineering process for developing control applications	Proposal	Process	SPEM 1.1	ERT	[181]
A SystemC-only design methodology and the CINE-IP multimedia platform	Proposal	Process	SPEM 1.1	ERT	[8]
Enhancing AUTOSAR methodology to a cotsbased development process via mapping to V-Model	Proposal	Process	SPEM 2.0	ERT	[103]
Towards a Specific Software Development Process for High Integrity Systems	Validation	Process	SPEM 2.0	ERT	[142]
A development framework for semantically interoperable health information systems.	Proposal	Process	SPEM 2.0	HCS	[114]
Integrating Instructional and Study Materials to Tailor a Student-Specific Resource	Validation	Process	SPEM 2.0	LR	[129]
Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies	Proposal	Process	SPEM 2.0 Extension	MAS	[170]
agentTool process editor	Proposal	Tool	UMA	MAS	[66]
Merging Agents and Services - the JIAC Agent Platform	Proposal	Framework	SPEM 1.1	MAS	[79]
Connecting methodologies and infrastructures in the development of agent systems	Philosophical	Process	SPEM 1.1	MAS	[28]
ANEMONA Development Process	Proposal	Process	SPEM 1.1	MAS	[23]
A Formal Description Language for Multi-Agent Architectures	Proposal	Process	SPEM 1.1	MAS	[57]
Methodology Fragments Definition in SPEM for Designing Adaptive Methodology: A First Step	Proposal	Process	SPEM 2.0	MAS	[155]
ASPECS: an agent-oriented software process for engineering complex systems	Proposal	Process	SPEM 2.0	MAS	[36]
Situated process engineering for integrating processes from methodologies to infrastructures	Proposal	Process	SPEM 2.0	MAS	[128]
A Software Engineering Guideline for Self-Organizing Resource-Flow Systems	Proposal	Process	SPEM 2.0	MAS	[168]
On the Modeling, Refinement and Integration of Decentralized Agent Coordination	Proposal	Process	SPEM 2.0	MAS	[178]
A technique fordefining agent-oriented engineering processes with tool support	Proposal	Technique	SPEM 2.0	MAS	[62]
Composition of a New Process to Meet Agile Needs Using Method Engineering	Proposal	Process	SPEM 1.1	MAS,ASD	[37]
MODAL: A SPEM Extension to Improve Co-design Process Models	Proposal	Model	SPEM 2.0 Extension	MDD	[98]

Tabla A.1: Publicaciones sobre el modelado de procesos (cont.)

Title	Research Type	Contribution Type	MetaModel Type	Modeling	Ref.
An Integrated Approach for Model Driven Process Modeling and Enactment	Proposal	Framework	SPEM 2.0	MDD	[117]
Towards an MDA-Based Development Methodology	Proposal	Process	SPEM 1.1	MDD	[67]
A Process Framework for the Successful Adoption of Model Driven Development	Proposal	Process	SPEM 1.1	MDD	[119]
Allowing End-users to Participate within Model-Driven Development Approaches	Proposal	Process	SPEM 2.0	MDD	[186]
Assessing 3-D Integrated Software Development Processes: A New Benchmark	Proposal	Model	SPEM 1.1 Extension	MPM	[110]
Multi-perspective Software Process Modeling	Proposal	Tool	SPEM 2.0 Extension	MPM	[91]
Towards Detailed Software Artifact Specification with SPEMArti	Proposal	Model	SPEM 2.0 Extension	MPM	[174]
A Meta-Method for Defining Software Engineering Methods	Proposal	Model	SPEM 2.0 Devised	MPM	[54]
ReuseTool-An extensible tool support for object-oriented framework reuse	Validation	Process	SPEM 2.0	OO	[135]
In search of the sweet spot: agile open collaborative corporate software development	Philosophical	Process	SPEM 1.1	OSSD,ASD	[179]
Managing open source contributions for software project sustainability	Proposal	Process	SPEM 1.1	OSSD, CM	[171]
Strategy to Improve Quality for Software Applications: A Process View	Validation	Framework	SPEM 1.1	PQI	[13]
An Architecture-Oriented Model-Driven Requirements Engineering Approach	Proposal	Process	SPEM 2.0	RE	[112]
Evaluating requirements modeling methods based on user perceptions: A family of experiments	Validation	Process	UMA	RE,UP	[82]
An engineering process for developing Secure Data Warehouses	Proposal	Framework	SPEM 1.1	SS, DW	[183]
Analysis of Secure Mobile Grid Systems: A systematic approach	Validation	Framework	SPEM 2.0	SS	[154]
Assessment methodology for software process improvement in small organizations	Proposal	Framework	SPEM 2.0	SPA	[145]
Towards Support Processes for Web Projects	Proposal	Process	SPEM 1.1	SPA	[14]
The Baseline: The Milestone of Software Product Lines for Expert Systems Automatic Development	Proposal	Framework	SPEM 1.1	SPL	[26]
Software Factories: Describing the Assembly Process	Proposal	Process	SPEM 1.1	SPL	[12]

Tabla A.1: Publicaciones sobre el modelado de procesos (cont.)

Title	Research Type	Contribution Type	MetaModel Type	Modeling	Ref.
Asset Recovery and Their Incorporation into Product Lines	Proposal	Process	SPEM 1.1	SPL	[97]
Using software product lines to manage model families in model-driven engineering	Proposal	Model	SPEM 2.0	SPL	[11]
Organizational Testing Management Maturity Model for a Software Product Line	Proposal	Process	SPEM 2.0	SPL	[105]
A Meta-Model for Requirements Engineering in System Family Context for Software Process Improvement Using CMMI	Proposal	Framework	SPEM 1.1	SPL,CMMI	[29]
A series of development methodologies for a variety of systems in Korea	Experience	Process	SPEM 2.0 Devised	SPL,ERT	[107]
Introducing systematic reuse in mainstream software process	Proposal	Process	SPEM 1.1	SPL,UP	[104]
Towards Method Engineering of Model-Driven User Interface Development	Philosophical	Framework	SPEM 2.0 Extension	UI	[176]
Architectural and Technological Variability in Rich Internet Applications	Proposal	Process	SPEM 1.1	UI	[124]
Introducing Graphic Designers in a Web Development Process	Proposal	Process	SPEM 1.1	UI	[187]
Ontological Traceability over the Unified Process	Proposal	Tool	SPEM 1.1	UP, CM	[133]
Extending the OpenUP/Basic Requirements Discipline to Specify Capacity Requirements	Evaluation	Process	SPEM 2.0	UP, HPS	[22]
RUP Extension For the Software Performance	Proposal	Process	UMA	UP, HPS	[40]
RUP Extension for the Development of Secure Systems	Proposal	Process	UMA	UP, SS	[39]
Extending RUP to develop fault tolerant software	Proposal	Process	UMA	UP, SS	[38]
Model Driven Design of Web Service Operations using Web Engineering Practices	Proposal	Framework	SPEM 1.1	WS	[156]
Providing Methodological Support to Incorporate Presentation Properties in the Development of Web Services	Proposal	Framework	SPEM 1.1	WS, UI	[157]
Technical Software Development Process in the XML Domain	Experience	Process	SPEM 2.0	XML	[197]

Tabla A.2: Publicaciones sobre la adaptabilidad de procesos

Title	Research Type	Contribution Type	MetaModel Type	Ref.
Deriving Project-Specific Processes from Process Line Architecture with Commonality and Variability	Proposal	Technique	SPEM 1.0 Extension	[190]
FlexSPMF: A Framework for Modelling and Learning Flexibility in Software Processes	Proposal	Framework	SPEM 2.0 Extension	[122]
Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms	Proposal	Model	SPEM 2.0 Extension	[121]
SPEM Extension with software process architectural concepts	Proposal	Model	SPEM 2.0 Extension	[6]
Process Aspect: Handling Crosscutting Concerns during Software Process Improvement	Proposal	Technique	SPEM 2.0	[116]
A Case Retrieval Method for Knowledge-Based Software Process Tailoring Using Structural Similarity	Proposal	Technique	SPEM 2.0	[89]
PIT-ProcessM: A Software Process Improvement Meta-Model	Proposal	Model	SPEM 2.0 Devised	[123]
Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach	Validation	Technique	SPEM 2.0	[7]
An MDE Approach to Software Process Tailoring	Validation	Technique	SPEM 2.0	[149]
Automatic Reuse of Process Patterns in Process Modeling	Proposal	Technique	SPEM 2.0 Devised	[180]

Tabla A.3: Publicaciones sobre la verificación y validación de procesos

Title	Research Type	Contribution Type	MetaModel Type	Ref.
Definition of an Executable SPEM 2.0	Proposal	Model	SPEM 2.0 Extension	[15]
Modeling and analysis of knowledge flows in software processes through the extension of the software process engineering metamodel	Proposal	Model	SPEM 2.0 Extension	[153]
Applying UML and software simulation for process definition, verification, and validation	Proposal	Framework	SPEM 1.1 Devised	[80]
agentTool process editor	Proposal	Tool	UMA	[66]
Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach	Validation	Technique	SPEM 2.0	[7]
FMESP: Framework for the modeling and evaluation of software processes	Proposal	Framework	SPEM 1.1	[64]
An Ontology Driven Approach to Software Project Enactment with a Supplier	Proposal	Mapping	SPEM 2.0	[111]
Software Process Model Blueprints	Proposal	Tool	SPEM 2.0	[4]
Developing a Simulation Model Using a SPEM-Based Process Model and Analytical Models	Proposal	Mapping	SPEM 1.1	[139]
Management and Education on the Case-Based Complex e-Business Systems Based On Agent Centric Ontology and Simulation Games	Experience	Mapping	SPEM 1.1 Extension	[81]
Intelligent Learning Environment for Software Engineering Processes	Proposal	Tool	SPEM 1.1	[195]
An approach to analyzing the software process change impact using process slicing and simulation	Proposal	Mapping	SPEM 2.0	[138]
NextMove: A Framework for Distributed Task Coordination	Proposal	Framework	SPEM 1.1 Devised	[118]
Formally Founded, Plan-based Enactment of Software Development Processes	Proposal	Technique	SPEM 2.0 Devised	[61]
Towards a tool-supported approach for collaborative process modeling and enactment	Proposal	Model	SPEM 2.0 Extension	[184]

Tabla A.4: Publicaciones sobre configuración y despliegue de procesos

Title	Research Type	Contribution Type	MetaModel Type	Ref.
eSPEM - A SPEM Extension for Enactable Behavior Modeling	Proposal	Model	SPEM 2.0 Extension	[52]
Definition of an Executable SPEM 2.0	Proposal	Model	SPEM 2.0 Extension	[15]
Achieving process modeling and execution through the combination of aspect and model-driven engineering approaches	Proposal	Model	SPEM 1.1 Devised	[16]
Process Definition and Project Tracking in Model Driven Engineering	Proposal	Model	SPEM 1.1 Devised	[148]
Transformation of BPMN subprocesses based in SPEM using QVT	Proposal	Mapping	SPEM 1.1	[42]
Supporting the SPEM with a UML Extended Workflow Meta-Model	Proposal	Mapping	SPEM 1.1	[41]
MD wizard - a model-driven framework for wizard-based modeling guidance in UML tools	Proposal	Framework	SPEM 2.0 Extension	[173]
Odyssey-CCS: A Change Control System Tailored to Software Reuse	Proposal	Tool	SPEM 1.1	[113]
An Integrated Approach for Model Driven Process Modeling and Enactment	Proposal	Framework	SPEM 2.0	[117]
An Approach to Method-Tool Coupling for Software Development	Proposal	Tool	SPEM 2.0	[88]
A Methodological Framework and Software Infrastructure for the Construction of Software Production Methods	Proposal	Framework	SPEM 2.0	[30]
A Service Based Development Environment on Web 2.0 Platforms	Proposal	Tool	SPEM 2.0	[106]
MDA Tool for Telecom Service Functional Design	Evaluation	Tool	SPEM 2.0	[2]
A Web-based process and process models to find and deliver information to improve the quality of flight software	Proposal	Tool	SPEM 1.1	[167]
Eclipse-based management system for process innovation & methodology enhancement	Proposal	Tool	SPEM 2.0	[196]
Lean Management of Software Processes and Factories Using Business Process Modeling Techniques	Proposal	Tool	SPEM 2.0	[19]

Tabla A.5: Publicaciones sobre la evaluación de procesos

Title	Research Type	Contribution Type	MetaModel Type	Ref.
Defining Software Process Model Constraints with rules using OWL and SWRL	Proposal	Mapping	SPEM 2.0	[152]
A deviation-tolerant approach to software process evolution	Validation	Tool	SPEM 1.1 Devised	[127]
The Use of a Meta-Model to Support Multi-Project Process Measurement	Proposal	Tool	SPEM 1.1	[32]

Anexo B

Metamodelos y reglas de transformación entre los modelos de productos software

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
→   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
 name="swpm" nsURI="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0"
 nsPrefix="swpm">
 <eClassifiers xsi:type="ecore:EClass" name="Project">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase contenedora de todos los productos de trabajo de un proyecto."
⇉   />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="workproducts" upperBound="-1" eType="#//WorkProduct"
   containment="true" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="WorkProduct"
  abstract="true">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase abstracta y similar al concepto de WorkProduct en \gls{SPEM}
    que representa un determinado producto generado o mantenido durante el
⇉   proceso software. Se distinguen entre productos entregables o internos." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Section">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase que representa una sección dentro de un determinado documento
⇉   generado durante el proyecto." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
```

```
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="order" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
⇉    EInt" />
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="embeddedSections" upperBound="-1" eType="#//Section"
    containment="true" />
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="models" upperBound="-1" eType="#//Model" containment="true" />
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="specifications" upperBound="-1" eType="#//Specification"
    containment="true" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Model">
   <eAnnotations source="comment">
    <details key="comment"
     value="Clase que representa un modelo diseñado haciendo uso un lenguaje de
⇉    modelado." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="type" eType="#//ModelType" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="description"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ModelType">
   <eAnnotations source="comment">
    <details key="comment"
     value="Enumeración con los posibles tipos de modelos. Comprende los ti-
     pos propuestos con el lenguaje UML, aunque podría englobar otros tipos de
     modelos visuales o textuales desarrollados con otros lenguajes, ya sean
⇉⇉    genéricos o de propósitos específicos (DSL)." />
   </eAnnotations>
   <eLiterals name="USECASE" literal="USECASE" />
   <eLiterals name="CLASS" value="1" literal="CLASS" />
   <eLiterals name="ACTIVITY" value="2" />
   <eLiterals name="COMPONENT" value="3" />
   <eLiterals name="DEPLOYMENT" value="4" />
   <eLiterals name="SEQUENCE" value="5" />
   <eLiterals name="STATECHART" value="6" />
   <eLiterals name="COLLABORATION" value="7" />
   <eLiterals name="OTHER" value="8" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Specification">
   <eAnnotations source="comment">
    <details key="comment"
```

```
    value="Clase que representa una especificación textual de algún aspecto del
⇉   proyecto software." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="type" eType="#//SpecificationType" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="CodeWorkProduct"
  eSuperTypes="#//InternalProduct">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase que representa la base de código de un proyecto software." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="packages" upperBound="-1" eType="#//SourcePackage" containment="true"
⇉   />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="DocumentaryWorkProduct"
  eSuperTypes="#//InternalProduct">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase que representa un documento técnico de trabajo." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="sections" upperBound="-1" eType="#//Section" containment="true" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="DeliverableProduct"
  eSuperTypes="#//WorkProduct">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase abstracta que representa un producto de trabajo desarrollado
    durante el transcurso de un proyecto software y con valor para terceros."
⇉   />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="embeddedWorkproducts" upperBound="-1" eType="#//WorkProduct" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="type" eType="#//DeliverableType" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="InternalProduct"
  eSuperTypes="#//WorkProduct">
  <eAnnotations source="comment">
   <details key="comment"
```

```
    value="Clase abstracta que representa un producto interno al desarrollo o
⇉   mantenimiento de software. Puede ser de tipo documental o de código." />
  </eAnnotations>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EEnum" name="DeliverableType">
  <eAnnotations source="comment">
   <details key="comment"
    value="Enumeración con los posibles tipos de entregables de un proyecto.
    Pueden ser ficheros empaquetados, instaladores o documentos, entre otros
⇉   tipos." />
  </eAnnotations>
  <eLiterals name="DOC" />
  <eLiterals name="SOURCE" value="1" />
  <eLiterals name="ARCHIVE" value="2" />
  <eLiterals name="INSTALLER" value="3" />
  <eLiterals name="PACKAGE" value="4" />
  <eLiterals name="OTHER" value="5" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="SourcePackage">
  <eAnnotations source="comment">
   <details key="comment"
    value="Clase que representa un determinado paquete de código fuente basado
⇉   en algún lenguaje de programación." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="type" eType="#//Archetype" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EEnum" name="Archetype">
  <eAnnotations source="comment">
   <details key="comment"
    value="Enumeración con los posibles tipos de paquetes de código fuente.
    Engloba a aquellas plantillas predefinidas de código, siguiendo los arque-
    tipos disponibles en sistemas de gestión de la construcción como Maven u
⇉⇉  otros." />
  </eAnnotations>
  <eLiterals name="J2EE" />
  <eLiterals name="STRUTS2" value="1" />
  <eLiterals name="SPRING_OSGI" value="2" />
  <eLiterals name="JSF" value="3" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EEnum" name="SpecificationType">
  <eAnnotations source="comment">
   <details key="comment"
```

```
    value="Enumeración con los posibles tipos de especificaciones no basadas en
    modelos. Pueden consistir en una descripción textual, una lista de elemen-
⇉   tos o contenido binario gestionado por alguna herramienta externa." />
  </eAnnotations>
  <eLiterals name="TEXTUAL" value="1" />
  <eLiterals name="EXTERNALCONTENT" value="2" />
  <eLiterals name="ITEMLIST" value="3" />
 </eClassifiers>
</ecore:EPackage>
```

Listado B.1: Metamodelo Ecore de SWPM

```
module uma2wspm;
create output: SWPM from input: UMA;

--- Matching MethodPlugin with Project
rule MethodPlugin2Project {
 from
  mp: UMA!MethodPlugin
 to
  proj: SWPM!Project (
   name <- mp.name,
   description <- mp.briefDescription,
   workproducts <- UMA!Domain.allInstances()
  )
}

--- Matching Deliverable with DeliverableProduct
rule Deliverable2DeliverableProduct {
 from
  del: UMA!Deliverable
 to
  dp: SWPM!DeliverableProduct (
   name <- del.presentationName,
   description <- del.briefDescription
  )
}

--- Matching Domain with DocumentaryWorkProduct and Section
rule Domain2DocumentaryWorkProductAndSection {
 from
  domain: UMA!Domain
 to
  dwp: SWPM!DocumentaryWorkProduct (
   name <- domain.presentationName,
```

```
    description <- domain.briefDescription,
    sections <- sec
   ),
   sec: SWPM!Section (
    name <- 'Default Section',
    order <- 0,
    description <- 'Container of models',
    models <- domain.workProducts
   )
}

--- Matching Artifact with Model
rule Artifact2Model {
 from
  art: UMA!Artifact (
   art.presentationName.size() > 0
  )
 to
  model: SWPM!Model (
   name <- art.presentationName,
   description <- art.briefDescription,
   type <- #CLASS
  )
}
```

Listado B.2: Reglas de transformación ATL entre UMA y SWPM

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
⇉  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
 name="wikim" nsURI="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0"
 nsPrefix="wikim">
 <eClassifiers xsi:type="ecore:EClass" name="Category"
  eSuperTypes="#//WikiContent">
  <eAnnotations source="comment">
   <details key="comment"
   value="Clase que representa a una categoría dentro una instancia de la
⇉  wiki." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="parentCategory" eType="#//Category" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="WikiContent">
```

```
    <eAnnotations source="comment">
     <details key="comment"
     value="Clase abstracta que representa a un tipo de contenido concreto que
⇒    puede almacenar una wiki." />
    </eAnnotations>
    <eOperations name="render"
     eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name"
     eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="Section">
    <eAnnotations source="comment">
     <details key="comment"
     value="Clase que permite declarar una sección dentro de un artículo de la
⇒    wiki." />
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="embeddedSections" upperBound="-1" eType="#//Section"
     containment="true" />
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name"
     eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="text"
     eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="contents" upperBound="-1" eType="#//SectionContent"
     containment="true" />
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="WikiDatabase">
    <eAnnotations source="comment">
     <details key="comment"
      value="Clase contenedora de todos los contenidos existentes en la wiki." />
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="contents" upperBound="-1" eType="#//WikiContent" containment="true" />
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="Article"
    eSuperTypes="#//WikiContent">
    <eAnnotations source="comment">
     <details key="comment"
      value="Clase que representa a un artículo dentro de la wiki." />
    </eAnnotations>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="sections" upperBound="-1" eType="#//Section" containment="true" />
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="categories" upperBound="-1" eType="#//Category" />
```

```
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Image"
   eSuperTypes="#//SectionContent">
   <eAnnotations source="comment">
    <details key="comment"
    value="Clase que representa la inclusión de una imagen dentro de un artícu-
⇉   lo wiki." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="uri"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="alternative"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SectionContent"
   abstract="true">
   <eAnnotations source="comment">
    <details key="comment"
    value="Clase abstracta que se especializa en cada tipo de contenido que
⇉   puede incluirse dentro de una sección de un artículo." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="order" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
⇉   EInt" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Paragraph"
   eSuperTypes="#//SectionContent">
   <eAnnotations source="comment">
    <details key="comment" value="Clase que representa un párrafo de texto." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="text"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ItemList"
   eSuperTypes="#//SectionContent">
   <eAnnotations source="comment">
    <details key="comment" value="Clase que representa un lista de items." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="items" upperBound="-1" eType="#//Item" containment="true" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Item">
   <eAnnotations source="comment">
```

```
    <details key="comment" value="Clase que representa el item de una lista." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="text"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="File"
   eSuperTypes="#//WikiContent">
   <eAnnotations source="comment">
    <details key="comment"
   value="Clase que representa a un determinado archivo binario almacenado en
⇉   la wiki." />
   </eAnnotations>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="User"
   eSuperTypes="#//WikiContent">
   <eAnnotations source="comment">
    <details key="comment"
   value="Clase que representa al usuario y la página del mismo dentro de la
⇉   wiki." />
   </eAnnotations>
  </eClassifiers>
 </ecore:EPackage>
```

Listado B.3: Metamodelo Ecore de WIKIM

```
module swpm2wikim;
create output: WIKIM from input: SWPM;

--- Matching Project with WikiDatabase and Category
rule Project2WikiDatabase {
 from
  proj: SWPM!Project
 to
  db: WIKIM!WikiDatabase (
   contents <- proj.workproducts,
   contents <- thisModule.obtainCategory(proj)
  )
}

unique lazy rule obtainCategory {
 from
  project: SWPM!Project
 to
```

```
    cat: WIKIM!Category (
     name <- project.name
    )
  }

  --- Matching DocumentaryWorkProduct with Article
  rule DocumentaryWorkProduct2Article {
   from
    doc: SWPM!DocumentaryWorkProduct
   to
    article: WIKIM!Article (
     name <- doc.name,
     sections <- doc.sections,
     categories <- thisModule.obtainCategory(doc.refImmediateComposite())
    )
  }

  --- Matching Section with Section
  rule Section2Section {
   from
    sourceSection: SWPM!Section
   to
    targetSection: WIKIM!Section (
     name <- sourceSection.name,
     text <- sourceSection.description,
     embeddedSections <- sourceSection.embeddedSections,
    embeddedSections <- sourceSection.models -> collect(model | thisModu-
⇉   le.resolveTemp(model, 'sec')).
     append(sourceSection.specifications -> collect(spec | thisModu-
⇉   le.resolveTemp(spec, 'sec')))
    )
  }

  --- Matching Model with Image
  rule Model2Image {
   from
    model: SWPM!Model
   to
    img: WIKIM!Image (
     uri <- thisModule.obtainElementType(model.type).img,
     alternative <- 'Model: '.concat(model.name),
     order <- thisModule.id
    ),
    item: WIKIM!Item (
    text <- 'Default '.concat(thisModule.obtainElementType(model.type).name).
⇉    concat(' for ').concat(model.name)
    ),
    items: WIKIM!ItemList (
```

```
   items <- item,
   name <- model.description,
   order <- img.order+2
  ),
  sec: WIKIM!Section (
   name <-model.name,
   contents <- img,
   contents <- items
  )
  do{
   thisModule.id<-thisModule.id+10;
  }

}

--- Matching Specification (item list) with Item List
rule ItemListSpecification2ItemList{
 from
  spec: SWPM!Specification(spec.type=#ITEMLIST)
 to
  item: WIKIM!Item (
   text <- spec.name.concat(' Default Item ')
  ),
  items: WIKIM!ItemList (
   items <- item,
   name <- spec.name
  ),
  sec: WIKIM!Section (
   name <-spec.name,
   contents <- items
  )
}

--- Matching Specification (textual) with Paragraph
rule TextualSpecification2Artifact{
 from
  spec: SWPM!Specification(spec.type=#TEXTUAL)
 to
  par: WIKIM!Paragraph (
   text <- spec.description
  ),
  sec: WIKIM!Section (
   name <-spec.name,
   contents <- par
  )
```

```
 }

 --- Counter
 helper def : id : Integer = 1;

 --- Auxiliary function
helper def: obtainElementType(diagram: String): TupleType(img: String, na-
⇉   me:String) =
  Map{
     (#CLASS,
     Tuple{img='http://spi-fm.uca.es/spdef/img/Logical.png',name='Class'}),
     (#USECASE,
     Tuple{img='http://spi-fm.uca.es/spdef/img/UseCase.png',name='Use Case'}),
     (#COMPONENT,
→   Tuple{img='http://spi-fm.uca.es/spdef/img/Component.png',name='Component'}),
     (#DEPLOYMENT,
     Tuple{img='http://spi-fm.uca.es/spdef/img/Deployment.png',name='Node'}),
     (#SEQUENCE,
     Tuple{img='http://spi-fm.uca.es/spdef/img/Sequence.png',name='Class'}),
     (#STATECHART,
     Tuple{img='http://spi-fm.uca.es/spdef/img/Statechart.png',name='State'}),
     (#COLLABORATION,
→   Tuple{img='http://spi-fm.uca.es/spdef/img/Collaboration.png',name='Class'}),
     (#ACTIVITY,
     Tuple{img='http://spi-fm.uca.es/spdef/img/Activity.png',name='Activity'})
     }.get(diagram);
```

Listado B.4: Reglas de transformación ATL entre SWPM y WIKIM

```
 <?xml version="1.0" encoding="UTF-8"?>
 <ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
⇉   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
  name="vmm" nsURI="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0"
  nsPrefix="vmm">
  <eClassifiers xsi:type="ecore:EClass" name="ModelRepository">
   <eAnnotations source="comment">
    <details key="comment"
   value="Clase contenedora de todos los proyectos gestionados con la herra-
⇉   mienta de modelado." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="projects" upperBound="-1" eType="#//Project" containment="true" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Package">
```

```
  <eAnnotations source="comment">
   <details key="comment"
   value="Clase que representa a un paquete de modelos, con el cual organizar
⇒  los diferentes modelos." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="diagrams" upperBound="-1" eType="#//Diagram" containment="true" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="embeddedPackages" upperBound="-1" eType="#//Package"
   containment="true" />
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="elements" upperBound="-1" eType="#//Element" containment="true" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Diagram">
  <eAnnotations source="comment">
   <details key="comment"
   value="Clase que representa a un determinado diagrama modelado con algún
⇒  lenguaje visual." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="type" eType="#//DiagramType" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="description"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedElements" upperBound="-1" eType="#//Element" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Element">
  <eAnnotations source="comment">
   <details key="comment"
   value="Enumeración con los posibles tipos de elementos que pueden parti-
   cipar en un diagrama. Casos de uso, clases o componentes son ejemplos de
⇒  tipos de elementos en modelos UML." />
  </eAnnotations>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="type" eType="#//ElementType" />
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="connectors" upperBound="-1" eType="#//Connector" containment="true" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
```

```
    name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="description"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Connector">
   <eAnnotations source="comment">
    <details key="comment"
    value="Clase que representa a una determinada relación entre dos elementos
⇉   de modelado." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="target" lowerBound="1" eType="#//Element" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="type" eType="#//ConnectorType" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="description"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Project">
   <eAnnotations source="comment">
    <details key="comment"
    value="Clase contenedora de todos los modelos generados para un determinado
⇉   proyecto." />
   </eAnnotations>
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="description"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="packages" upperBound="-1" eType="#//Package" containment="true" />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ElementType">
   <eAnnotations source="comment">
    <details key="comment"
    value="Enumeración con los posibles tipos de diagramas reconocidos por
    la herramienta de modelado, como por ejemplo los diagramas de clases y de
⇉   estados de UML." />
   </eAnnotations>
   <eLiterals name="CLASS" />
   <eLiterals name="USECASE" value="1" />
   <eLiterals name="COMPONENT" value="2" />
```

```
  <eLiterals name="NODE" value="3" />
  <eLiterals name="OBJECT" value="4" />
  <eLiterals name="STATE" value="5" />
  <eLiterals name="ACTIVITY" value="6" />
  <eLiterals name="ARTIFACT" value="7" />
  <eLiterals name="REQUIREMENT" value="8" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EEnum" name="ConnectorType">
  <eAnnotations source="comment">
   <details key="comment"
   value="Enumeración con los posibles tipos de conectores que pueden vincu-
   lar dos elementos, como por ejemplo las asociaciones, generalizaciones y
   relaciones de inclusión o extensión en UML." />
  </eAnnotations>
  <eLiterals name="ASSOCIATION" />
  <eLiterals name="AGGREGATION" value="1" />
  <eLiterals name="COMPOSITION" value="2" />
  <eLiterals name="DEPENDENCY" value="3" />
  <eLiterals name="INCLUDE" value="4" />
  <eLiterals name="EXTEND" value="5" />
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EEnum" name="DiagramType">
  <eAnnotations source="comment">
   <details key="comment"
   value="Enumeración con los posibles tipos de diagramas reconocidos por
   la herramienta de modelado, como por ejemplo los diagramas de clases y de
   estados de UML." />
  </eAnnotations>
  <eLiterals name="USECASE" />
  <eLiterals name="CLASS" value="1" />
  <eLiterals name="ACTIVITY" value="2" />
  <eLiterals name="COMPONENT" value="3" />
  <eLiterals name="DEPLOYMENT" value="4" />
  <eLiterals name="SEQUENCE" value="5" />
  <eLiterals name="STATECHART" value="6" />
  <eLiterals name="COLLABORATION" value="7" />
  <eLiterals name="OTHER" value="8" />
 </eClassifiers>
</ecore:EPackage>
```

Listado B.5: Metamodelo Ecore de VMM

```
module swpm2vmm;
create output: VMM from input: SWPM;
```

```
--- Matching Project with ModelRepository
rule Project2ModelRepository {
 from
  proj: SWPM!Project
 to
  db: VMM!ModelRepository(
   projects <- thisModule.Project2Project(proj)
  )
}


--- Matching Project with Project
lazy rule Project2Project {
 from
  proj: SWPM!Project
 to
  db: VMM!Project (
   packages <- proj.workproducts,
   name <- proj.name,
   description <- proj.description

  )
}

--- Matching DocumentaryWorkProducts with Package
rule DocumentaryWorkProduct2Package {
 from
  doc: SWPM!DocumentaryWorkProduct
 to
  pack: VMM!Package (
   name <- doc.name,
   description <- doc.description,
   embeddedPackages <- doc.sections
  )
}

--- Matching Section with Package
rule Section2Package {
 from
  sec: SWPM!Section
 to
  pack: VMM!Package (
   name <- sec.name,
   description <- sec.description,
   embeddedPackages <- sec.embeddedSections,
  diagrams <- sec.models -> collect(model | thisModu-
⇉  le.resolveTemp(model,'diag')),
```

```
   elements <- sec.models -> collect(model | thisModu-
⇒   le.resolveTemp(model,'element')),
   elements <- sec.specifications -> collect(model | thisModu-
⇒   le.resolveTemp(model,'element'))
   )
 }

 --- Matching Model with Diagram and Element
 rule Model2DiagramAndElement {
  from
   model: SWPM!Model
  to
   element: VMM!Element (
   name <- 'Default '.concat(thisModule.obtainElementType(model.type)).concat('
⇒   for ').concat(model.name),
   description <- 'Example description of the '.concat(thisModule. obtainEle-
⇒   mentType(model.type)),
    type <- thisModule.obtainElementType(model.type)
   ),
   diag: VMM!Diagram (
    name <- model.name,
    description <- model.description,
    type <- model.type,
    containedElements <- element
   )
 }

 --- Matching Specification (item list) with Element
 rule ItemListSpecification2Element{
  from
   spec: SWPM!Specification(spec.type=#ITEMLIST)
  to
   element: VMM!Element (
    name <- spec.name.concat(' Default Item '),
    description <- spec.description,
    type <- #REQUIREMENT
   )
 }

 --- Matching Specification (textual) with Artifact
 rule TextualSpecification2Element{
  from
   spec: SWPM!Specification(spec.type=#TEXTUAL)
  to
   element: VMM!Element (
    name <- spec.name,
    description <- spec.description,
    type <- #ARTIFACT
```

```
  )
}

--- Auxiliary function
helper def: obtainElementType(diagram: String): String =
 Map{
    (#CLASS,
    'CLASS'),
    (#USECASE,
    'USECASE'),
    (#COMPONENT,
    'COMPONENT'),
    (#DEPLOYMENT,
    'NODE'),
    (#SEQUENCE,
    'CLASS'),
    (#STATECHART,
    'STATE'),
    (#COLLABORATION,
    'CLASS'),
    (#ACTIVITY,
    'ACTIVITY')}.get(diagram);
```

Listado B.6: Reglas de transformación ATL entre SWPM y VMM

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
⇉  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
 name="mwm" nsURI="http://spi-fm.uca.es/spdef/models/specificTools/mwm/1.0"
 nsPrefix="mwm">
 <eClassifiers xsi:type="ecore:EClass" name="MediaWikiDatabase">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="wikiDatabase"
  eType="ecore:EClass ../../genericTools/wikim/WikitoolModel.ecore#// WikiDa-
⇉  tabase" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="url"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="login"
   eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="password"
```

```
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
 </ecore:EPackage>
```

Listado B.7: Metamodelo Ecore de MediaWiki

```
 <?xml version="1.0" encoding="UTF-8"?>
 <ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
⇉   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
  name="eam" nsURI="http://spi-fm.uca.es/spdef/models/specificTools/eam/1.0"
  nsPrefix="eam">
  <eClassifiers xsi:type="ecore:EClass" name="EnterpriseArchitectDatabase">
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="modelRepository"
   eType="ecore:EClass ../../genericTools/vmm/VisualModelingtoolModel.ecore#//
⇉   ModelRepository" />
   <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="fileName"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" />
  </eClassifiers>
 </ecore:EPackage>
```

Listado B.8: Metamodelo Ecore de Enterprise Architect

Anexo C

Vocabularios y mappings entre los modelos de productos software

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
        <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;" xmlns:xsd="&xsd;">
 <rdfs:Class
rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Documen-
⇉  taryWorkProduct">
  <rdfs:subClassOf
 rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Inter-
⇉  nalProduct" />
  <rdfs:label>The DocumentaryWorkProduct entity</rdfs:label>
  <rdfs:comment>Clase que representa un documento técnico de trabajo.
  </rdfs:comment>
 </rdfs:Class>
 <rdf:Property
rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# workpro-
⇉  ducts">
  <rdfs:label>The workproducts reference</rdfs:label>
  <rdfs:domain
 rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Project"
⇉  />
  <rdfs:range
 rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Work-
⇉  Product" />
 </rdf:Property>

 <rdfs:Class
rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# SourcePac-
⇉  kage">
  <rdfs:label>The SourcePackage entity</rdfs:label>
  <rdfs:comment>Clase que representa un determinado paquete de código
   fuente basado en algún lenguaje de programación.</rdfs:comment>
 </rdfs:Class>
 <rdf:Property
→rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#description">
  <rdfs:label>The description attribute</rdfs:label>
  <rdfs:domain
 rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Project"
⇉  />
  <rdfs:domain
 rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Work-
⇉  Product" />
  <rdfs:domain
 rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇉  />
  <rdfs:domain
```

```
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Model"
⇉   />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Speci-
⇉   fication" />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Source-
⇉   Package" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
   <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#order">
    <rdfs:label>The order attribute</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇉   />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int" />
   </rdf:Property>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Model">
    <rdfs:label>The Model entity</rdfs:label>
    <rdfs:comment>Clase que representa un modelo diseñado haciendo uso un
     lenguaje de modelado.</rdfs:comment>
   </rdfs:Class>
   <rdfs:Class
  rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# CodeWork-
⇉   Product">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Inter-
⇉   nalProduct" />
    <rdfs:label>The CodeWorkProduct entity</rdfs:label>
    <rdfs:comment>Clase que representa la base de código de un proyecto
     software.</rdfs:comment>
   </rdfs:Class>
   <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#packages">
    <rdfs:label>The packages reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Code-
⇉   WorkProduct" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Source-
⇉   Package" />
   </rdf:Property>
   <rdfs:Class
→ rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#WorkProduct">
    <rdfs:label>The WorkProduct entity</rdfs:label>
    <rdfs:comment>Clase abstracta y similar al concepto de WorkProduct en
     SPEM que representa un determinado producto generado o
     mantenido durante el proceso software. Se distinguen entre productos
```

```
    entregables o internos.</rdfs:comment>
  </rdfs:Class>
  <rdfs:Class
 rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Specifica-
⇒  tion">
   <rdfs:label>The Specification entity</rdfs:label>
   <rdfs:comment>Clase que representa una especificación textual de algún
    aspecto del proyecto software.</rdfs:comment>
  </rdfs:Class>
  <rdf:Property
   rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#name">
   <rdfs:label>The name attribute</rdfs:label>
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Project"
⇒   />
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Work-
⇒   Product" />
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇒   />
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Model"
⇒   />
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Speci-
⇒   fication" />
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Source-
⇒   Package" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </rdf:Property>
  <rdf:Property
   rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#models">
   <rdfs:label>The models reference</rdfs:label>
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇒   />
   <rdfs:range
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Model"
⇒   />
  </rdf:Property>
  <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#int">
  </rdfs:Datatype>
  <rdf:Property
   rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#sections">
   <rdfs:label>The sections reference</rdfs:label>
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Docu-
⇒   mentaryWorkProduct" />
```

```
    <rdfs:range
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇉    />
   </rdf:Property>
   <rdfs:Class
  rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Internal-
⇉    Product">
    <rdfs:subClassOf
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Work-
⇉    Product" />
    <rdfs:label>The InternalProduct entity</rdfs:label>
    <rdfs:comment>Clase abstracta que representa un producto interno al
     desarrollo o mantenimiento de software. Puede ser de tipo documental
     o de código.</rdfs:comment>
   </rdfs:Class>
   <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#string">
   </rdfs:Datatype>
   <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# embedded-
⇉    Sections">
    <rdfs:label>The embeddedSections reference</rdfs:label>
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇉    />
    <rdfs:range
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇉    />
   </rdf:Property>
   <rdfs:Class
  rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Delivera-
⇉    bleProduct">
    <rdfs:subClassOf
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Work-
⇉    Product" />
    <rdfs:label>The DeliverableProduct entity</rdfs:label>
    <rdfs:comment>Clase abstracta que representa un producto de trabajo
     desarrollado durante el transcurso de un proyecto software y con
     valor para terceros.</rdfs:comment>
   </rdfs:Class>
   <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#type">
    <rdfs:label>The type attribute</rdfs:label>
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Model"
⇉    />
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Speci-
⇉    fication" />
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Delive-
⇉    rableProduct" />
```

```
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Source-
⇉   Package" />
    <rdfs:range
     rdf:resource="http://www.w3.org/2004/02/skos/core#Concept" />
  </rdf:Property>
  <rdfs:Class
   rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Project">
   <rdfs:label>The Project entity</rdfs:label>
   <rdfs:comment>Clase contenedora de todos los productos de trabajo de
    un proyecto.</rdfs:comment>
  </rdfs:Class>
  <rdfs:Class
   rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section">
   <rdfs:label>The Section entity</rdfs:label>
   <rdfs:comment>Clase que representa una sección dentro de un
    determinado documento generado durante el proyecto.</rdfs:comment>
  </rdfs:Class>
  <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# specifica-
⇉   tions">
   <rdfs:label>The specifications reference</rdfs:label>
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Section"
⇉   />
   <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Speci-
⇉   fication" />
  </rdf:Property>
  <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# embedded-
⇉   Workproducts">
   <rdfs:label>The embeddedWorkproducts reference</rdfs:label>
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Delive-
⇉   rableProduct" />
   <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Work-
⇉   Product" />
  </rdf:Property>
 </rdf:RDF>
```

Listado C.1: Vocabulario RDF Schema de SWPM

```
<http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Project>
 rdfs:subClassOf <http://usefulinc.com/ns/doap#Project> .
```

```
<http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#name>
 rdfs:subPropertyOf <http://purl.org/dc/elements/1.1/title> .

<http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#description>
 rdfs:subPropertyOf <http://purl.org/dc/elements/1.1/description> .
```

Listado C.2: Mappings del vocabulario SWPM

```
<http://spi-fm.uca.es/spdef/kos/Archetype>
 rdf:type  <http://www.w3.org/2004/02/skos/core#ConceptScheme> ;
 rdfs:label "ARCHETYPE";
 rdfs:comment "Enumeración con los posibles tipos de paquetes de código
   fuente. Engloba a aquellas plantillas predefinidas de código,
   siguiendo los arquetipos disponibles en sistemas de gestión de la
   construcción como Maven u otros." .

<http://spi-fm.uca.es/spdef/kos/Archetype#J2EE>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "J2EE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/Archetype> .

<http://spi-fm.uca.es/spdef/kos/Archetype#Struts2>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "STRUTS2";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/Archetype> .

<http://spi-fm.uca.es/spdef/kos/Archetype#Spring_OSGI>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "SPRING_OSGI";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/Archetype> .

<http://spi-fm.uca.es/spdef/kos/Archetype#JSF>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "JSF";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/Archetype> .

<http://spi-fm.uca.es/spdef/kos/ModelType>
 rdf:type  <http://www.w3.org/2004/02/skos/core#ConceptScheme> ;
 rdfs:label "MODELTYPE";
 rdfs:comment "Enumeración con los posibles tipos de modelos. Comprende
   los tipos propuestos con el lenguaje UML, aunque podría englobar
   otros tipos de modelos visuales o textuales desarrollados con otros
   lenguajes, ya sean genéricos o de propósitos específicos (DSL)" .
```

```
<http://spi-fm.uca.es/spdef/kos/ModelType#UseCase>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "USECASE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Class>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "CLASS";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Activity>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "ACTIVITY";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Component>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "COMPONENT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Deployment>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "DEPLOYMENT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Sequence>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "SEQUENCE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#StateChart>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "STATECHART";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Collaboration>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "COLLABORATION";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/ModelType#Other>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "OTHER";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ModelType> .

<http://spi-fm.uca.es/spdef/kos/SpecificationType>
```

```
 rdf:type  <http://www.w3.org/2004/02/skos/core#ConceptScheme> ;
 rdfs:label "SPECIFICATIONTYPE";
 rdfs:comment "Enumeración con los posibles tipos de especificaciones
   no basadas en modelos. Pueden consistir en una descripción textual,
   una lista de elementos o contenido binario gestionado por alguna
   herramienta externa." .

<http://spi-fm.uca.es/spdef/kos/SpecificationType#Textual>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "TEXTUAL";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/SpecificationType> .

<http://spi-fm.uca.es/spdef/kos/SpecificationType#ExternalContent>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "EXTERNALCONTENT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/SpecificationType> .

<http://spi-fm.uca.es/spdef/kos/SpecificationType#ItemList>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "ITEMLIST";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/SpecificationType> .

<http://spi-fm.uca.es/spdef/kos/DeliverableType>
 rdf:type  <http://www.w3.org/2004/02/skos/core#ConceptScheme> ;
 rdfs:label "DELIVERABLETYPE";
 rdfs:comment "Enumeración con los posibles tipos de entregables de un
   proyecto. Pueden ser ficheros empaquetados, instaladores o
   documentos, entre otros tipos" .

<http://spi-fm.uca.es/spdef/kos/DeliverableType#Doc>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "DOC";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/DeliverableType> .

<http://spi-fm.uca.es/spdef/kos/DeliverableType#Source>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "SOURCE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/DeliverableType> .

<http://spi-fm.uca.es/spdef/kos/DeliverableType#Archive>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "ARCHIVE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/DeliverableType> .

<http://spi-fm.uca.es/spdef/kos/DeliverableType#Installer>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "INSTALLER";
```

```
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/DeliverableType> .

<http://spi-fm.uca.es/spdef/kos/DeliverableType#Package>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "PACKAGE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/DeliverableType> .

<http://spi-fm.uca.es/spdef/kos/DeliverableType#Other>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "OTHER";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/DeliverableType> .
```

Listado C.3: Esquema de clasificación SKOS para el vocabulario SWPM

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
        <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;" xmlns:xsd="&xsd;">
 <rdfs:Class
 rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# WikiDa-
⇉  tabase">
  <rdfs:label>The WikiDatabase entity</rdfs:label>
  <rdfs:comment>Clase contenedora de todos los contenidos existentes en
   la wiki.</rdfs:comment>
 </rdfs:Class>
 <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#text">
  <rdfs:label>The text attribute</rdfs:label>
  <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉  Section" />
  <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉  Paragraph" />
  <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Item"
⇉  />
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
 </rdf:Property>
 <rdfs:Class
 rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# Section-
⇉  Content">
```

```
    <rdfs:label>The SectionContent entity</rdfs:label>
    <rdfs:comment>Clase abstracta que se especializa en cada tipo de
     contenido que puede incluirse dentro de una sección de un artículo.
    </rdfs:comment>
   </rdfs:Class>
   <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# alterna-
⇉   tive">
    <rdfs:label>The alternative attribute</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Image"
⇉   />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
   <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#items">
    <rdfs:label>The items reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   ItemList" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Item"
⇉   />
   </rdf:Property>
   <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#order">
    <rdfs:label>The order attribute</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   SectionContent" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int" />
   </rdf:Property>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#User">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   WikiContent" />
    <rdfs:label>The User entity</rdfs:label>
    <rdfs:comment>Clase que representa al usuario y la página del mismo
     dentro de la wiki.</rdfs:comment>
   </rdfs:Class>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Item">
    <rdfs:label>The Item entity</rdfs:label>
    <rdfs:comment>Clase que representa el item de una lista.
    </rdfs:comment>
   </rdfs:Class>
   <rdf:Property
→ rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#contents">
```

```
    <rdfs:label>The contents reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   Section" />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   WikiDatabase" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   SectionContent" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   WikiContent" />
   </rdf:Property>
   <rdfs:Class
→ rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Category">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   WikiContent" />
    <rdfs:label>The Category entity</rdfs:label>
    <rdfs:comment>Clase que representa a una categoría dentro una
     instancia de la wiki.</rdfs:comment>
   </rdfs:Class>
   <rdfs:Class
→ rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#ItemList">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   SectionContent" />
    <rdfs:label>The ItemList entity</rdfs:label>
    <rdfs:comment>Clase que representa un lista de items.</rdfs:comment>
   </rdfs:Class>
   <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#int">
   </rdfs:Datatype>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#File">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   WikiContent" />
    <rdfs:label>The File entity</rdfs:label>
    <rdfs:comment>Clase que representa a un determinado archivo binario
     almacenado en la wiki.</rdfs:comment>
   </rdfs:Class>
   <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#string">
   </rdfs:Datatype>
   <rdfs:Class
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# WikiCon-
⇉   tent">
    <rdfs:label>The WikiContent entity</rdfs:label>
    <rdfs:comment>Clase abstracta que representa a un tipo de contenido
     concreto que puede almacenar una wiki.</rdfs:comment>
```

```
  </rdfs:Class>
  <rdfs:Class
→ rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Paragraph">
   <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   SectionContent" />
   <rdfs:label>The Paragraph entity</rdfs:label>
   <rdfs:comment>Clase que representa un párrafo de texto.</rdfs:comment>
  </rdfs:Class>
  <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# embed-
⇉   dedSections">
   <rdfs:label>The embeddedSections reference</rdfs:label>
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   Section" />
   <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   Section" />
  </rdf:Property>
  <rdf:Property
→ rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#sections">
   <rdfs:label>The sections reference</rdfs:label>
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   Article" />
   <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   Section" />
  </rdf:Property>
  <rdf:Property
   rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#name">
   <rdfs:label>The name attribute</rdfs:label>
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   WikiContent" />
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   Section" />
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇉   ItemList" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </rdf:Property>
  <rdf:Property
   rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#uri">
   <rdfs:label>The uri attribute</rdfs:label>
   <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Image"
⇉   />
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Article">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇒   WikiContent" />
    <rdfs:label>The Article entity</rdfs:label>
    <rdfs:comment>Clase que representa a un artículo dentro de la wiki.
    </rdfs:comment>
   </rdfs:Class>
   <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# parent-
⇒   Category">
    <rdfs:label>The parentCategory reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇒   Category" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇒   Category" />
   </rdf:Property>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Section">
    <rdfs:label>The Section entity</rdfs:label>
    <rdfs:comment>Clase que permite declarar una sección dentro de un
     artículo de la wiki.</rdfs:comment>
   </rdfs:Class>
   <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0# catego-
⇒   ries">
    <rdfs:label>The categories reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇒   Article" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇒   Category" />
   </rdf:Property>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Image">
    <rdfs:subClassOf
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#
⇒   SectionContent" />
    <rdfs:label>The Image entity</rdfs:label>
    <rdfs:comment>Clase que representa la inclusión de una imagen dentro
     de un artículo wiki.</rdfs:comment>
   </rdfs:Class>
  </rdf:RDF>
```

Listado C.4: Vocabulario RDF Schema de WIKIM

```
<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#User>
 rdfs:subClassOf <http://xmlns.com/foaf/0.1/Person> .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Category>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Pro-
⇉   ject> .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Article>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Docu-
⇉   mentaryWorkProduct> .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Section>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Sec-
⇉   tion> .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#Image>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0# Model>
⇉   .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#name>
rdfs:subPropertyOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#
⇉   name>  .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#description>
rdfs:subPropertyOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#
⇉   description>  .

<http://spi-fm.uca.es/spdef/models/genericTools/wikim/1.0#sections>
rdfs:subPropertyOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#
⇉   sections>  .
```

Listado C.5: Mappings del vocabulario WIKIM

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
        <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;" xmlns:xsd="&xsd;">
 <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#packages">
  <rdfs:label>The packages reference</rdfs:label>
```

```
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Project"
⇉   />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉   />
  </rdf:Property>
  <rdf:Property
 rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# containe-
⇉   dElements">
    <rdfs:label>The containedElements reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram"
⇉   />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉   />
  </rdf:Property>
  <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Connector">
    <rdfs:label>The Connector entity</rdfs:label>
    <rdfs:comment>Clase que representa a una determinada relación entre
     dos elementos de modelado.</rdfs:comment>
  </rdfs:Class>
  <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#target">
    <rdfs:label>The target reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# Con-
⇉    nector" />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉   />
  </rdf:Property>
  <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#type">
    <rdfs:label>The type attribute</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram"
⇉   />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉   />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# Con-
⇉    nector" />
    <rdfs:range
     rdf:resource="http://www.w3.org/2004/02/skos/core#Concept" />
  </rdf:Property>
  <rdf:Property
```

```
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#name">
    <rdfs:label>The name attribute</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉   />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram"
⇉   />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉   />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# Con-
⇉    nector" />
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Project"
⇉   />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </rdf:Property>
  <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package">
    <rdfs:label>The Package entity</rdfs:label>
    <rdfs:comment>Clase que representa a un paquete de modelos, con el
     cual organizar los diferentes modelos.</rdfs:comment>
  </rdfs:Class>
  <rdf:Property
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# embedded-
⇉   Packages">
    <rdfs:label>The embeddedPackages reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉   />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉   />
  </rdf:Property>
  <rdf:Property
→ rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#connectors">
    <rdfs:label>The connectors reference</rdfs:label>
    <rdfs:domain
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉   />
    <rdfs:range
   rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# Con-
⇉    nector" />
  </rdf:Property>
  <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#elements">
    <rdfs:label>The elements reference</rdfs:label>
    <rdfs:domain
```

```
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉    />
     <rdfs:range
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉    />
   </rdf:Property>
   <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#string">
   </rdfs:Datatype>
   <rdfs:Class
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram">
    <rdfs:label>The Diagram entity</rdfs:label>
    <rdfs:comment>Clase que representa a un determinado diagrama modelado
     con algún lenguaje visual.</rdfs:comment>
   </rdfs:Class>
   <rdf:Property
→rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#description">
    <rdfs:label>The description attribute</rdfs:label>
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉    />
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram"
⇉    />
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element"
⇉    />
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# Con-
⇉    nector" />
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Project"
⇉    />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
   </rdf:Property>
   <rdf:Property
    rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#projects">
    <rdfs:label>The projects reference</rdfs:label>
    <rdfs:domain
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# Model-
⇉    Repository" />
    <rdfs:range
    rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Project"
⇉    />
   </rdf:Property>
   <rdfs:Class
  rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0# ModelRepo-
⇉    sitory">
    <rdfs:label>The ModelRepository entity</rdfs:label>
    <rdfs:comment>Clase contenedora de todos los proyectos gestionados con
     la herramienta de modelado.</rdfs:comment>
```

```
  </rdfs:Class>
  <rdf:Property
   rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#diagrams">
   <rdfs:label>The diagrams reference</rdfs:label>
   <rdfs:domain
  rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Package"
⇉   />
   <rdfs:range
  rdf:resource="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram"
⇉   />
  </rdf:Property>
  <rdfs:Class
   rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Project">
   <rdfs:label>The Project entity</rdfs:label>
   <rdfs:comment>Clase contenedora de todos los modelos generados para un
    determinado proyecto.</rdfs:comment>
  </rdfs:Class>
  <rdfs:Class
   rdf:about="http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Element">
   <rdfs:label>The Element entity</rdfs:label>
   <rdfs:comment>Enumeración con los posibles tipos de elementos que
    pueden participar en un diagrama. Casos de uso, clases o componentes
    son ejemplos de tipos de elementos en modelos UML.</rdfs:comment>
  </rdfs:Class>
 </rdf:RDF>
```

Listado C.6: Vocabulario RDF Schema de VMM

```
 <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Project>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Project>
⇉ .

 <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#Diagram>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#Model>
⇉ .

 <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#name>
rdfs:subPropertyOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#name>
⇉ .

 <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#description>
 rdfs:subPropertyOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#
⇉   description> .

 <http://spi-fm.uca.es/spdef/models/genericTools/vmm/1.0#diagrams>
rdfs:subClassOf <http://spi-fm.uca.es/spdef/models/deployment/swpm/1.0#models>
⇉ .
```

Listado C.7: Mappings del vocabulario VMM

```
<http://spi-fm.uca.es/spdef/kos/ElementType>
 rdf:type  <http://www.w3.org/2004/02/skos/core#ConceptScheme> ;
 rdfs:label "ELEMENTTYPE";
 rdfs:comment "Enumeración con los posibles tipos de diagramas
   reconocidos por la herramienta de modelado, como por ejemplo los
   diagramas de clases y de estados de UML." .

<http://spi-fm.uca.es/spdef/kos/ElementType#Class>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "CLASS";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

 <http://spi-fm.uca.es/spdef/kos/ElementType#UseCase>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "USECASE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#Component>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "COMPONENT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#Node>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "NODE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#Object>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "OBJECT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#State>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "STATE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#Activity>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "ACTIVITY";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#Artifact>
```

```
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "ARTIFACT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ElementType#Requirement>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "REQUIREMENT";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ElementType> .

<http://spi-fm.uca.es/spdef/kos/ConnectorType>
 rdf:type  <http://www.w3.org/2004/02/skos/core#ConceptScheme> ;
 rdfs:label "CONNECTORTYPE";
 rdfs:comment "Enumeración con los posibles tipos de conectores que
   pueden vincular dos elementos, como por ejemplo las asociaciones,
   generalizaciones y relaciones de inclusión o extensión en UML." .

 <http://spi-fm.uca.es/spdef/kos/ConnectorType#Association>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "ASSOCIATION";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ConnectorType> .

 <http://spi-fm.uca.es/spdef/kos/ConnectorType#Aggregation>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "AGGREGATION";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ConnectorType> .

 <http://spi-fm.uca.es/spdef/kos/ConnectorType#Composition>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "COMPOSITION";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ConnectorType> .

 <http://spi-fm.uca.es/spdef/kos/ConnectorType#Dependency>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "DEPENDENCY";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ConnectorType> .

 <http://spi-fm.uca.es/spdef/kos/ConnectorType#Include>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "INCLUDE";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ConnectorType> .

 <http://spi-fm.uca.es/spdef/kos/ConnectorType#Extend>
 rdf:type  <http://www.w3.org/2004/02/skos/core#Concept> ;
 rdfs:label "EXTEND";
 skos:inScheme <http://spi-fm.uca.es/spdef/kos/ConnectorType> .
```

Listado C.8: Esquema de clasificación SKOS para el vocabulario VMM

Glosario de siglas

ALM Application Lifecycle Management. 55, 56, 68
API Application Programming Interface. 77, 109, 113
AS Automotive Systems. 42, 165
ASD Agile Software Development. 42, 165–167
ATL Atlas Transformation Language. 64, 94, 106, 107, 109, 118, 120, 142, 143, 158

BPEL Business Process Execution Language. 44, 62
BPM Business Process Management. 30, 31, 36, 37, 45, 47, 53
BPMN Business Process Modeling Notation. 14, 27, 53, 54, 62, 64, 158

CaS Context-aware Systems. 42, 165
CIM Computation Independent Model. 62, 80, 102, 106, 159
CM Change Management. 42, 165, 167, 168
CMMI Capability Maturity Model Integration. 14, 41, 42, 44, 47, 146, 157, 165, 168
CRM Customer Relationship Management Systems. 42, 165
CWM Common Warehouse Metamodel. 63

DOAP Description Of A Project. 67, 110
DSL Domain Specific Language. 62, 86, 145
DW Data Warehouse. 42, 165, 167

EII Enterprise Information Integration. 65, 79, 132
EPF Eclipse Process Framework. 24, 30, 45, 107, 118, 158
ERT Embedded Real-Time Systems. 42, 165, 166, 168

ETL Extract, Transform and Load. 65, 79, 128

HCS Health Care Systems. 42, 166
HPS High Performance Systems. 42, 168
HTTP Hypertext Transfer Protocol. 69, 112

ISO International Organization for Standardization. 3, 68
ITM Issue Tracking tool Model. 89, 93, 94, 110, 112, 137

JCR Journal Citation Reports. 148–151
JPDL JBPM Process Definition Language. 14, 44
JSON JavaScript Object Notation. 112

LMF Linked Media Framework. 132, 133
LOD Linked Open Data. 5, 7, 59, 65, 66, 68, 69, 71, 76–80, 94, 110, 112, 113, 115, 132, 137, 142, 150, 151, 154
LR Learning Resources. 42, 166

M2M Model-to-Model. 35, 63, 64, 74, 109
M2T Model-to-Text. 35, 64, 75
MAD Mobile Application Development. 42, 165
MAS Multi-Agent Systems. 41, 42, 166
MDA Model-Driven Architecture. 45, 49, 62–64, 74, 76
MDD Model-Driven Development. 42, 61, 71, 166, 167
MDE Model-Driven Engineering. 5, 7, 27, 29, 59–61, 74, 80, 94, 102, 120, 142
MOF Meta-Object Facility. 15, 47, 63, 64
MOFM2T MOF Model to Text Transformation Language. 63
MPM Multi-Perspective Process Modeling. 42, 167
MVC Model View Controller. 77, 112, 144

OCL Object Constraint Language. 44, 46, 48, 63, 64
OMG Object Management Group. 15, 19, 36, 59, 60, 62, 63, 141
OO Object-oriented. 42, 167
OSSD Open Source Software Development. 42, 167
OWL Web Ontology Language. 68, 132

PIM Platform Independent Model. 62, 80, 81, 106, 110, 144, 145, 159

PQI Product Quality Improvement. 42, 167
PSM Platform Specific Model. 62, 80, 85, 91, 106, 110, 145, 159

RDF Resource Description Framework. 66–69, 77, 79, 80, 110–113, 131–133, 142–145, 147, 153, 154, 158
RE Requirements Engineering. 42, 167
REST Representational State Transfer. 69, 77, 79

SJR SCImago Journal Rank. 148, 152, 153
SKOS Simple Knowledge Organization System. 111, 112
SME Situational Method Engineering. 29, 43
SMS Systematic Mapping Study. 7, 158
SOFTECO Software Ecosystems. 42, 165
SPA Software Process Assessment. 42, 167
SPARQL SPARQL Protocol and RDF Query Language. 69, 79, 112, 131–134, 137, 142, 154, 159
SPCM Software Project Control Model. 84, 94, 96, 100, 106, 110, 144
SPDEF Software Process Deployment & Evaluation Framework. 71, 141
SPDT Software Process Deployment Toolkit. 107, 145
SPEM Software and Systems Process Engineering Metamodel Specification. 6, 13, 15, 18–22, 24, 25, 27, 29–31, 33–36, 38–41, 44–49, 55, 63, 73–75, 80, 81, 83, 94, 96, 98, 100, 102, 106, 141, 142, 144, 148, 158
SPL Software Product Lines. 42, 43, 47, 167, 168
SS Secure Systems. 42, 167, 168
SWPM Software Work Product Model. 81, 94, 96, 98, 104, 106, 107, 110–113, 118, 137, 144

UI User Interface. 42, 168
UMA Unified Method Architecture. 24, 102, 106, 107, 118
UML Unified Modeling Language. 14, 15, 22, 24, 25, 27, 44, 47, 48, 54, 62–64, 73, 80, 83, 88, 89, 91, 93, 106, 107, 109, 113, 118, 120, 134, 135, 137
UP Unified Process. 41, 42, 46, 142, 167, 168
URI Uniform Resource Identifier. 112, 133, 134

VMM Visual Modeling tool Model. 88, 93, 94, 107, 110–112, 120, 137

W3C World Wide Web Consortium. 66, 68, 111
WIKIM WIKI tool Model. 85, 91, 94, 107, 110, 112
WS Web Services. 42, 165, 168

XMI XML Metadata Interchange. 63, 107, 158
XML eXtensible Markup Language. 41, 42, 67, 112, 168
XPDL XML Process Definition Language. 14, 44

Referencias

[1] Abad, Z.S.H., Sadi, M., Ramsin, R.: Towards Tool Support for Situational Engineering of Agile Methodologies. In: 2010 Asia Pacific Software Engineering Conference, pp. 326–335. IEEE (2010)

[2] Ahuja, A., Simonin, J., Nedelec, R.: MDA Tool for Telecom Service Functional Design. Software Architecture pp. 519–522 (2010)

[3] Al-asmari, K.R., Yu, L.: Experiences in Distributed Software Development with Wiki. Software Engineering Research and Practice (2006)

[4] Alegría, J., Lagos, A., Bergel, A., Bastarrica, M.: Software Process Model Blueprints. In: J. Münch, Y. Yang, W. Schäfer (eds.) New Modeling Concepts for Today's Software Processes, *Lecture Notes in Computer Science*, vol. 6195, pp. 273–284. Springer Berlin / Heidelberg (2010)

[5] Allemang, D., Hendler, J.: Semantic web for the working ontologist: effective modeling in RDFS and OWL. Elsevier (2011)

[6] Aoussat, F. and Oussalah, M. and Nacer, M.A.: SPEM Extension with software process architectural concepts. In: Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual. IEEE (2011)

[7] Ara, F., Freire, A., Grande, R.: Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach. In: Enterprise Information Systems 12th International Conference, ICEIS 2010,, pp. 372–387. Springer (2011)

[8] Araújo, G., Barros, E., Melcher, E., Azevedo, R., Silva, K., Prado, B., Lima, M.: A SystemC-only design methodology and the CINE-IP multimedia platform. Design Automation for Embedded Systems **10**(2), 181–202 (2005)

[9] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., et al.: Gene ontology: tool for the unification of biology. Nature genetics **25**(1), 25–29 (2000)

[10] Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. Software, IEEE **20**(5), 36–41 (2003)

[11] Avila-García, O., García, A.E., Rebull, E.V.S.: Using software product lines to manage model families in model-driven engineering. In: Proceedings of the 2007 ACM symposium on Applied computing - SAC '07, SAC '07, p. 1006. ACM Press, New York, New York, USA (2007)

[12] Azanza, M., Díaz, O., Trujillo, S.: Software Factories: Describing the Assembly Process. In: J. Münch, Y. Yang, W. Schäfer (eds.) New Modeling Concepts for Today's Software Processes, *Lecture Notes in Computer Science*, vol. 6195, pp. 126–137. Springer Berlin / Heidelberg (2010)

[13] Becker, P., Lew, P., Olsina, L.: Strategy to Improve Quality for Software Applications: A Process View. In: Proceeding of the 2nd workshop on Software engineering for sensor network applications, pp. 129–138 (2011)

[14] Becker, P., Olsina, L.: Towards Support Processes for Web Projects. In: F. Daniel, F. Facca (eds.) Current Trends in Web Engineering, *Lecture Notes in Computer Science*, vol. 6385, pp. 102–113. Springer Berlin / Heidelberg (2010)

[15] Bendraou, R., Combemale, B., Cregut, X., Gervais, M.P.: Definition of an Executable SPEM 2.0. In: 14th Asia-Pacific Software Engineering Conference (APSEC'07), pp. 390–397. IEEE (2007)

[16] Bendraou, R., Jezequel, J.M., Fleurey, F.: Achieving process modeling and execution through the combination of aspect and model-driven engineering approaches. Journal of Software Maintenance and Evolution: Research and Practice pp. n/a–n/a (2010)

[17] Bendraou, R., Jezequel, J.M., Gervais, M.P., Blanc, X.: A Comparison of Six UML-Based Languages for Software Process Modeling. IEEE Transactions on Software Engineering **36**(5), 662–675 (2010)

[18] Bendraou, R., Jézéquel, J.M., Gervais, M.P., Blanc, X.: A comparison of six uml-based languages for software process modeling. Software Engineering, IEEE Transactions on **36**(5), 662–675 (2010)

[19] Berrocal, J., García-Alonso, J., Murillo, J.: Lean Management of Software Processes and Factories Using Business Process Modeling Techniques. In: M. Ali Babar, M. Vierimaa, M. Oivo (eds.) Product-Focused Software Process Improvement, *Lecture Notes in Computer Science*, vol. 6156, pp. 321–335. Springer Berlin / Heidelberg (2010)

[20] Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) **5**(3), 1–22 (2009)

[21] Boehm, B.: Foundations of empirical software engineering: the legacy of Victor R. Basili. Springer (2005)

[22] Borg, A., Sandahl, K., Patel, M.: Extending the OpenUP/Basic Requirements Discipline to Specify Capacity Requirements. In: 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 328–333. IEEE (2007)

[23] Botti, V., Giret, A.: ANEMONA Development Process. In: ANEMONA, Springer Series in Advanced Manufacturing, pp. 91–133. Springer London (2008)

[24] Bourque, P., Dupuis, R.: Guide to the Software Engineering Body of Knowledge 2004 Version. Software Engineering Body of Knowledge 2004 SWEBOK, Guide to the (2004)

[25] Brambilia, M., Cabot, J., Wimmer, M.: Model-driven Software Engineering in Practice, vol. 1. Morgan & Claypool Publishers (2012)

[26] Cabello, M.E., Ramos, I.: The Baseline: The Milestone of Software Product Lines for Expert Systems Automatic Development. In: 2008 Mexican International Conference on Computer Science, pp. 44–51. IEEE (2008)

[27] Cabot, J., Wilson, G., et al.: Tools for teams: A survey of web-based software project portals. Dr. Dobb's pp. 1–14 (2009)

[28] Cabri, G., Puviani, M., Quitadamo, R.: Connecting methodologies and infrastructures in the development of agent systems. In: 2008 International Multiconference on Computer Science and Information Technology, pp. 17–23. IEEE (2008)

[29] Cerón, R., Dueñas, J.C., Serrano, E., Capilla, R.: A Meta-model for Requirements Engineering in System Family Context for Software Process Improvement Using CMMI. In: F. Bomarius, S. Komi-Sirviö (eds.) Product-Focused Software Process Improvement, *Lecture Notes in Computer Science*, vol. 3547, pp. 291–313. Springer Berlin / Heidelberg (2005)

[30] Cervera, M., Albert, M., Torres, V., Pelechano, V.: A Methodological Framework and Software Infrastructure for the Construction of Software Production Methods. In: J. Münch, Y. Yang, W. Schäfer (eds.) New Modeling Concepts for Today's Software Processes, *Lecture Notes in Computer Science*, vol. 6195, pp. 112–125. Springer Berlin / Heidelberg (2010)

[31] Chen, N.: Convention over configuration (2006). URL http://softwareengineering.vazexqi.com/files/pattern.html

[32] Colombo, A., Damiani, E., Frati, F., Oltolina, S., Reed, K., Ruffatti, G.: The Use of a Meta-Model to Support Multi-Project Process Measurement. In: 2008 15th Asia-Pacific Software Engineering Conference, pp. 503–510. IEEE (2008)

[33] Commision, E.: European e-competence framework version 2.0. URL http://www.ecompetences.eu/

[34] Committee, I.S.C.: IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos. CA: IEEE Computer Society **121990** (1990)

[35] Cossentino, M., Garro, A.: Activity of the FIPA Methodology Technical Committee. ICAR-CNR Technical Report RT-ICAR-PA-05-15 (2005)

[36] Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: ASPECS: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems **20**(2), 260–304 (2010)

[37] Cossentino, M., Seidita, V.: Composition of a New Process to Meet Agile Needs Using Method Engineering. In: Software Engineering for Multi-Agent Systems III, *Lecture Notes in Computer Science*, vol. 3390, pp. 36–51. Springer Berlin / Heidelberg (2005)

[38] de B. Paes, C.E., Hirata, C.M., Yano, E.T.: Extending RUP to develop fault tolerant software. In: Proceedings of the 2008 ACM symposium on Applied computing - SAC '08, SAC '08, p. 783. ACM Press, New York, New York, USA (2008)

[39] de Barros Paes, C.E., Hirata, C.M.: RUP Extension for the Development of Secure Systems. In: Fourth International Conference on Information Technology (ITNG'07), pp. 643–652. IEEE (2007)

[40] de Barros Paes, C.E., Hirata, C.M.: RUP Extension For the Software Performance. In: 2008 32nd Annual IEEE International Computer Software and Applications Conference, pp. 732–738. IEEE (2008)

[41] Debnath, N., Riesco, D., Cota, M.P., Garcia Perez-Schofield, J., Uva, D.: Supporting the SPEM with a UML Extended Workflow Metamodel. In: IEEE International Conference on Computer Systems and Applications, 2006., pp. 1151–1154. IEEE (2006)

[42] Debnath, N., Zorzan, F.A., Montejano, G., Riesco, D.: Transformation of BPMN subprocesses based in SPEM using QVT. In: 2007 IEEE International Conference on Electro/Information Technology, pp. 146–151. IEEE (2007)

[43] Decker, B., Ras, E., Rech, J., Jaubert, P., Rieth, M.: Wiki-Based Stakeholder Participation in Requirements Engineering. IEEE Software (2007)

[44] DeMarco, T.: Controlling software projects: Management, measurement, and estimates. Prentice Hall PTR Upper Saddle River, NJ, USA (1986)

[45] Deming, W.E.: Out of the crisis, 1986. Cambridge, Mass.: Massachusetts Institute of Technology Center for Advanced Engineering Study. xiii **507** (1986)

[46] Díaz, O., Puente, G.: Wiki scaffolding: Aligning wikis with the corporate strategy. Information Systems **37**(8), 737–752 (2012)

[47] Dodero, J.M., Rodríguez, G., Ibarra, M.S.: Análisis de las contribuciones a un wiki para la evaluación web de competencias. In: Actas de la Conferencia Conjunta Iberoamericana sobre Tecnologías de Aprendizaje, pp. 268–277 (2009)

[48] Dodero, J.M., Ruiz-Rube, I., Palomo-Duarte, M., Cabot, J.: Model-driven learning design. Journal of Research and Practice in Information Technology **44**(3), 61 (2012)

[49] Dodero, J.M., Ruiz-Rube, I., Palomo-Duarte, M., Vázquez-Murga, J.: Open linked data model revelation and access for analytical web science. In: Metadata and Semantic Research, pp. 105–116. Springer (2011)

[50] Dodero, J.M., del Val, Á.M., Torres, J.: An extensible approach to visually editing adaptive learning activities and designs based on services. Journal of Visual Languages & Computing **21**(6), 332–346 (2010)

[51] Dolado, J.J., Fernández, L., et al.: Medición para la gestión en la ingeniería del software. Editorial RA-MA, ISBN pp. 84–7897 (2000)

[52] Ellner, R., Al-Hilank, S., Drexler, J., Jung, M., Kips, D., Philippsen, M.: eSPEM – A SPEM Extension for Enactable Behavior Modeling. In: T. Kühne, B. Selic, M.P. Gervais, F. Terrier (eds.) Modelling Foundations and Applications, *Lecture Notes in Computer Science*, vol. 6138, pp. 116–131. Springer Berlin / Heidelberg (2010)

[53] Emami, M.S., Ithnin, N.B., Ibrahim, O.: Software process engineering: Strengths, weaknesses, opportunities and threats. In: Networked Computing (INC), 2010 6th International Conference on, pp. 1–5. IEEE (2010)

[54] Engels, G., Sauer, S.: A Meta-Method for Defining Software Engineering Methods. In: G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, B. Westfechtel (eds.) Graph Transformations and Model-Driven Engineering, *Lecture Notes in Computer Science*, vol. 5765, pp. 411–440. Springer Berlin / Heidelberg (2010)

[55] Escalona, M.J., Koch, N.: Metamodeling the requirements of web systems. In: Web Information Systems and Technologies, pp. 267–280. Springer (2007)

[56] Falcone Sampaio, P.: Business Process Design and Implementation for Customer Segmentation e-Services. In: 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, pp. 228–234. IEEE (2005)

[57] Faulkner, S., Kolp, M., Wautelet, Y., Achbany, Y.: A Formal Description Language for Multi-Agent Architectures. In: M. Kolp, B. Henderson-Sellers, H. Mouratidis, A. Garcia, A. Ghose, P. Bresciani (eds.) Agent-Oriented Information Systems IV, *Lecture Notes in Computer Science*, vol. 4898, pp. 143–163. Springer Berlin / Heidelberg (2008)

[58] Fenton, N.E., Pfleeger, S.L.: Software metrics: a rigorous and practical approach. PWS Publishing Co. (1998)

[59] Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California (2000)

[60] Firesmith, D.G., Henderson-Sellers, B.: The OPEN process framework: An introduction. Addison-Wesley Professional (2002)

[61] Friedrich, J., Bergner, K.: Formally Founded, Plan-based Enactment of Software Development Processes. In: Proceeding of the 2nd workshop on Software engineering for sensor network applications, pp. 199–203. ACM (2011)

[62] Fuentes-Fernández, R., García-Magariño, I., Gómez-Rodríguez, A., González-Moreno, J.: A technique fordefining agent-oriented engineering processes with tool support. Engineering Applications of Artificial Intelligence **23**(3), 432–444 (2010)

[63] Fuggetta, A.: Software process: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, pp. 25–34. ACM (2000)

[64] García, F., Piattini, M., Ruiz, F., Canfora, G., VISAGGIO, C.: FMESP: Framework for the modeling and evaluation of software processes. Journal of Systems Architecture **52**(11), 627–639 (2006)

[65] García-Borgoñón, L., Barcelona, M., García-García, J., Alba, M., Escalona, M.: Software process modeling languages: a systematic literature review. Information and Software Technology (2013)

[66] Garcia-Ojeda, J.C., DeLoach, S.A.: agentTool process editor. In: Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09, SAC '09, p. 707. ACM Press, New York, New York, USA (2009)

[67] Gavras, A., Belaunde, M., Pires, L.F., Almeida, J.a.P.A.: Towards an MDA-Based Development Methodology. In: F. Oquendo, B. Warboys, R. Morrison (eds.) Software Architecture, *Lecture Notes in Computer Science*, vol. 3047, pp. 230–240. Springer Berlin / Heidelberg (2004)

[68] Gašević, D., Devedžić, V., Djuriâc, D.: Model driven engineering and ontology development. Springer (2009)

[69] Glorio, O., Mazón, J.n., Garrigós, I., Trujillo, J.: A personalization process for spatial data warehouse development. Decision Support Systems pp. 1–15 (2012)

[70] Gonzalo Valdes, Hernán Astudillo, M.V., López, C.: The Tutelkan Reference Process: A Reusable Process Model for Enabling SPI in Small Settings. In: Systems, Software and Services Process Improvement 17th European Conference, EuroSPI 2010,, pp. 179–190. Springer (2011)

[71] Gronback, R.C.: Eclipse modeling project: a domain-specific language (DSL) toolkit. Pearson Education (2009)

[72] Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? International journal of human-computer studies **43**(5), 907–928 (1995)

[73] Gruber, T.R., et al.: A translation approach to portable ontology specifications. Knowledge acquisition **5**(2), 199–220 (1993)

[74] Halevy, A.Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., Sikka, V.: Enterprise information integration: successes, challenges and controversies. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 778–787. ACM (2005)

[75] Heath, T., Bizer, C.: Linked data: Evolving the web into a global data space. Synthesis lectures on the semantic web: theory and technology **1**(1), 1–136 (2011)

[76] Henderson-Sellers, B., Ralyt, J.: Situational Method Engineering: State-of-the-Art Review. Journal of Universal Computer Science **16**(3), 424–478 (2010)

[77] Herraiz, I., Gonzalez-Barahona, J.M., Robles, G., German, D.M.: On the prediction of the evolution of libre software projects. 2007 IEEE International Conference on Software Maintenance pp. 405–414 (2007)

[78] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly **28**(1), 75–105 (2004)

[79] Hirsch, B., Konnerth, T., Hessler, A.: Merging Agents and Services — the JIAC Agent Platform. In: A. El Fallah Seghrouchni, J. Dix, M. Dastani, R.H. Bordini (eds.) Multi-Agent Programming:, pp. 159–185. Springer US (2009)

[80] Hsueh, N., Shen, W., Yang, Z., Yang, D.: Applying UML and software simulation for process definition, verification, and validation. Information and Software Technology **50**(9-10), 897–911 (2008)

[81] Huang, S.T., Hsu, M.C., Lin, W.H.: Management and Education on the Case-Based Complex e-Business Systems Based On Agent Centric Ontology and Simulation Games. In: IEEE International Conference on e-Business Engineering (ICEBE'07), pp. 379–382. IEEE (2007)

[82] Insfran, E., Genero, M.: Evaluating requirements modeling methods based on user perceptions: A family of experiments. Information Sciences **181**, 3356–3378 (2011)

[83] ISO: ISO/IEC 24744. Tech. rep., International Organization for Standardization/International Electrotechnical Commission and others (2007)

[84] Jeong, Y.J., Lee, J.H., Shin, G.S.: Development Process of Mobile Application SW Based on Agile Methodology. In: 2008 10th International Conference on Advanced Communication Technology, pp. 362–366. IEEE (2008)

[85] Joerg, B., Ruiz-Rube, I., Sicilia, M.A., Dvořvoák, J., Jeffery, K., Hoellrigl, T., Rasmussen, H.S., Engfer, A., Vestdam, T., Barriocanal, E.G.: Connecting closed world research information systems through the linked open data web. International Journal of Software Engineering and Knowledge Engineering **22**(03), 345–364 (2012)

[86] Joksimovic, S., Jovanovic, J., Gasevic, D., Zouaq, A., Jeremic, Z.: An empirical evaluation of ontology-based semantic annotators. In: Proceedings of the seventh international conference on Knowledge capture, pp. 109–112. ACM (2013)

[87] Juran, J.: Quality in Software Development. McGraw-Hill Professional (1998)

[88] Kaindl, H., Falb, J., Melbinger, S., Bruckmayer, T.: An Approach to Method-Tool Coupling for Software Development. In: 2010 Fifth International Conference on Software Engineering Advances, pp. 101–106. IEEE (2010)

[89] Kang, D., Song, I.G., Park, S., Bae, D.H., Kim, H.K., Lee, N.: A Case Retrieval Method for Knowledge-Based Software Process Tailoring Using

Structural Similarity. In: 2008 15th Asia-Pacific Software Engineering Conference, pp. 51–58. IEEE (2008)

[90] Kelleher, J.: A reusable traceability framework using patterns. In: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering - TEFSE '05, TEFSE '05, p. 50. ACM Press, New York, New York, USA (2005)

[91] Kerzazi, N., Robillard, P.N.: Multi-perspective Software Process Modeling. In: 2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications, pp. 85–92. IEEE (2010)

[92] Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. Journal of the ACM (JACM) **42**(4), 741–843 (1995)

[93] Kimball, R., Caserta, J.: The data warehouse ETL toolkit. John Wiley & Sons (2004)

[94] Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Engineering (2007)

[95] Kleppe, A., Warmer, J., Bast, W.: The Model Driven Architecture: Practice and Promise. 2003. Addison Wesley (1995)

[96] Klyne, G., Carroll, J.J., McBride, B.: Resource description framework (rdf): Concepts and abstract syntax. W3C recommendation **10** (2004)

[97] Knodel, J., John, I., Ganesan, D., Pinzger, M., Usero, F., Arciniegas, J., Riva, C.: Asset Recovery and Their Incorporation into Product Lines. In: 12th Working Conference on Reverse Engineering (WCRE'05), pp. 120–129. IEEE (2005)

[98] Koudri, A., Champeau, J.: MODAL: A SPEM Extension to Improve Codesign Process Models. In: J. Münch, Y. Yang, W. Schäfer (eds.) New Modeling Concepts for Today's Software Processes, *Lecture Notes in Computer Science*, vol. 6195, pp. 248–259. Springer Berlin / Heidelberg (2010)

[99] Krammer, M. and Armengaud, E. and Bourrouilh, Q.: Method Library Framework for Safety Standard Compliant Process Tailoring. In: Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on, pp. 302–305. IEEE (2011)

[100] Krasteva, I., Ilieva, S., Dimov, A.: Experience-Based Approach for Adoption of Agile Practices in Software Development Projects. In: B. Pernici (ed.) Advanced Information Systems Engineering, *Lecture Notes in Computer Science*, vol. 6051, pp. 266–280. Springer Berlin / Heidelberg (2010)

[101] Kruchten, P.: The rational unified process: an introduction. Addison-Wesley Professional (2004)

[102] Kulkarni, V., Reddy, S.: Introducing MDA in a large IT consultancy organization. In: Software Engineering Conference, 2006. APSEC 2006. 13th Asia Pacific, pp. 419–426 (2006)

[103] Kumar, M., Yoo, J., Hong, S.: Enhancing AUTOSAR methodology to a cotsbased development process via mapping to V-Model. In: 2009 IEEE International Symposium on Industrial Embedded Systems, pp. 50–53. IEEE (2009)

[104] Laguna, M., Gonzalez-Baixauli, B., Lopez, O., Garcia, F.: Introducing systematic reuse in mainstream software process. In: Proceedings of the 20th IEEE Instrumentation Technology Conference (Cat No 03CH37412) EURMIC-03, pp. 351–358. IEEE (2003)

[105] Lamas, E., Ferreira, E., Ribeiro do Nascimento, M., Dias, L.A.V., Silveira, F.F.: Organizational Testing Management Maturity Model for a Software Product Line. In: 2010 Seventh International Conference on Information Technology: New Generations, pp. 1026–1031. IEEE (2010)

[106] Larrucea, X., Fernandez, R., Soriano, J., Martínez, A., Gonzalez-Barahona, J.M.: A Service Based Development Environment on Web 2.0 Platforms. In: P. Mähönen, K. Pohl, T. Priol (eds.) Towards a Service-Based Internet, *Lecture Notes in Computer Science*, vol. 5377, pp. 38–48. Springer Berlin / Heidelberg (2008)

[107] Lee, J., Kim, J.S., Cho, J.H.: A series of development methodologies for a variety of systems in Korea. In: Proceeding of the 28th international conference on Software engineering - ICSE '06, ICSE '06, p. 612. ACM Press, New York, New York, USA (2006)

[108] Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 233–246. ACM (2002)

[109] Li, J., Li, M., Wu, Z., Wang, Q.: A Metamodel for the CMM Software Process. In: J. Cao, L. Yang, M. Guo, F. Lau (eds.) Parallel and Distributed Processing and Applications, *Lecture Notes in Computer Science*, vol. 3358, pp. 446–450. Springer Berlin / Heidelberg (2005)

[110] Li, M.: Assessing 3-D Integrated Software Development Processes: A New Benchmark. In: Q. Wang, D. Pfahl, D. Raffo, P. Wernick (eds.) Software Process Change, *Lecture Notes in Computer Science*, vol. 3966, pp. 15–38. Springer Berlin / Heidelberg (2006)

[111] Líška, M., Návrat, P.: An Ontology Driven Approach to Software Project Enactment with a Supplier. In: B. Catania, M. Ivanovic, B. Thalheim (eds.) Advances in Databases and Information Systems, *Lecture Notes in Computer Science*, vol. 6295, pp. 378–391. Springer Berlin / Heidelberg (2011)

[112] Loniewski, G., Armesto, A., Insfran, E.: An Architecture-Oriented Model-Driven Requirements Engineering Approach. In: Model-Driven Requirements Engineering Workshop (MoDRE), 2011, Cim, pp. 31–38. IEEE (2011)

[113] Lopes, L., Murta, L., Werner, C.: Odyssey-CCS: A Change Control System Tailored to Software Reuse. In: M. Morisio (ed.) Reuse of Off-the-Shelf Components, *Lecture Notes in Computer Science*, vol. 4039, pp. 170–183. Springer Berlin / Heidelberg (2006)

[114] Lopez, D.M., Blobel, B.: A development framework for semantically interoperable health information systems. International journal of medical informatics **78**(2), 83–103 (2009)

[115] Louridas, P.: Using Wikis in Software Development. IEEE Software pp. 6–9 (2006)

[116] Ma, J.k., Shi, L., Wang, Y.s., Mei, H.: Process Aspect: Handling Crosscutting Concerns during Software Process Improvement. In: Q. Wang, V. Garousi, R. Madachy, D. Pfahl (eds.) Trustworthy Software Development Processes, *Lecture Notes in Computer Science*, vol. 5543, pp. 124–135. Springer Berlin / Heidelberg (2009)

[117] Maciel, R.S.P., da Silva, B.C., Magalhães, A.P.F., Rosa, N.S.: An Integrated Approach for Model Driven Process Modeling and Enactment. In:

2009 XXIII Brazilian Symposium on Software Engineering, pp. 104–114. IEEE (2009)

[118] Mak, D.K., Kruchten, P.B.: NextMove: A Framework for Distributed Task Coordination. In: 2007 Australian Software Engineering Conference (ASWEC'07), pp. 399–408. IEEE (2007)

[119] Mansell, J., Bediaga, A., Vogel, R., Mantell, K.: A Process Framework for the Successful Adoption of Model Driven Development. In: A. Rensink, J. Warmer (eds.) Model Driven Architecture – Foundations and Applications, *Lecture Notes in Computer Science*, vol. 4066, pp. 90–100. Springer Berlin / Heidelberg (2006)

[120] Martinez, P., Amescua, A., García, J., Cuadra, D., Llorens, J., Fuentes, J., Martin, D., Cuevas, G., Calvo-Manzano, J., San Feliu, T.: Requirements for a knowledge management framework to be used in software intensive organizations. In: IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005., pp. 554–559. IEEE (2005)

[121] Martínez-Ruiz, T., García, F., Piattini, M.: Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. In: R. Lee (ed.) Software Engineering Research, Management and Applications, *Studies in Computational Intelligence*, vol. 150, pp. 115–130. Springer Berlin / Heidelberg (2008)

[122] Martinho, R., Varajão, J.a., Domingos, D.: FlexSPMF: A Framework for Modelling and Learning Flexibility in Software Processes. In: M. Lytras, E. Damiani, J. Carroll, R. Tennyson, D. Avison, A. Naeve, A. Dale, P. Lefrere, F. Tan, J. Sipior, G. Vossen (eds.) Visioning and Engineering the Knowledge Society. A Web Science Perspective, *Lecture Notes in Computer Science*, vol. 5736, pp. 78–87. Springer Berlin / Heidelberg (2009)

[123] Martins, P.V., da Silva, A.R.: PIT-ProcessM: A Software Process Improvement Meta-Model. In: 2010 Seventh International Conference on the Quality of Information and Communications Technology, pp. 453–458. IEEE (2010)

[124] Meliá, S., Gómez, J., Pérez, S., Díaz, O.: Architectural and Technological Variability in Rich Internet Applications. IEEE Internet Computing **14**(3), 24–32 (2010)

[125] Menzel, C., Mayer, R.J.: The idef family of languages. In: Handbook on architectures of information systems, pp. 209–241. Springer (1998)

[126] Meyer, B.: Towards empirical answers to the core problems of software engineering. http://cacm.acm.org/blogs/blog-cacm/166529 (2013)

[127] Mohammed, K., Redouane, L., Bernard, C.: A deviation-tolerant approach to software process evolution. In: Ninth international workshop on Principles of software evolution in conjunction with the 6th ESEC/FSE joint meeting - IWPSE '07, IWPSE '07, p. 75. ACM Press, New York, New York, USA (2007)

[128] Molesini, A., Denti, E., Nardini, E., Omicini, A.: Situated process engineering for integrating processes from methodologies to infrastructures. In: Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09, SAC '09, p. 699. ACM Press, New York, New York, USA (2009)

[129] Monteith, J.Y.: Integrating Instructional and Study Materials to Tailor a Student- Specific Resource. In: Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on, pp. 294–303. IEEE (2011)

[130] Navarro, E., Adviser-Hoek, A.V.: Simse: a software engineering simulation environment for software process education. California State University at Long Beach (2006)

[131] Neto, R.F.B.a., Kudo, T., da Graça Pimentel, M.: Using a software process for ontology-based context-aware computing. In: Proceedings of the 12th Brazilian symposium on Multimedia and the web - WebMedia '06, WebMedia '06, p. 61. ACM Press, New York, New York, USA (2006)

[132] Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001, pp. 2–9. ACM (2001)

[133] Noll, R.P., Ribeiro, M.B.: Ontological Traceability over the Unified Process. In: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07), pp. 249–255. IEEE (2007)

[134] Oates, B.J.: Researching information systems and computing. Sage (2005)

[135] Oliveira, T.C., Alencar, P., Cowan, D.: ReuseTool-An extensible tool support for object-oriented framework reuse. The Journal of Systems & Software **84**(12), 2234–2252 (2011)

[136] OMG: Software & Systems Process Engineering Meta-Model Specification. Tech. Rep. April, Object Management Group (2008)

[137] Palomo-Duarte, M., Dodero, J.M., Medina-Bulo, I., Rodríguez-Posada, E.J., Ruiz-Rube, I.: Assessment of collaborative learning experiences by graphical analysis of wiki contributions. Interactive Learning Environments **0**(0), 1–23 (2012)

[138] Park, S., Bae, D.H.: An approach to analyzing the software process change impact using process slicing and simulation. Journal of Systems and Software **84**(4), 528–543 (2011)

[139] Park, S., Kim, H., Kang, D., Bae, D.H.: Developing a Simulation Model Using a SPEM-Based Process Model and Analytical Models. In: W. Aalst, J. Mylopoulos, N.M. Sadeh, M.J. Shaw, C. Szyperski, J.L.G. Dietz, A. Albani, J. Barjis (eds.) Advances in Enterprise Engineering I, *Lecture Notes in Business Information Processing*, vol. 10, pp. 164–178. Springer Berlin Heidelberg (2008)

[140] Peach, R.W.: The ISO 9000 handbook. Irwin Professional Publishing (1995)

[141] Pérez, J.: Notaciones y lenguajes de procesos. Una visión global. (2007)

[142] Perseil, I.: Towards a Specific Software Development Process for High Integrity Systems. ACM SIGSOFT Software Engineering Notes **36**(1), 1–8 (2011)

[143] Petersen, K., Feldt, R.: Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10. The Institution of Engineering and Technology (2008)

[144] Pettersson, O., Svensson, M., Gil, D., Andersson, J., Milrad, M.: On the role of software process modeling in software ecosystem design. In: Proceedings of the Fourth European Conference on Software Architecture Companion Volume - ECSA '10, ECSA '10, p. 103. ACM Press, New York, New York, USA (2010)

[145] Pino, F.J., Pardo, C., García, F., Piattini, M.: Assessment methodology for software process improvement in small organizations. Information and Software Technology **52**(10), 1044–1061 (2010)

[146] Pino, F.J., Ruiz, F., Piattini, M.: A software maintenance methodology for small organizations: Agile MANTEMA. Journal of Software Maintenance and Evolution: Research and Practice (2011)

[147] Pol, K., Patil, N., Patankar, S., Das, C.: A survey on web content mining and extraction of structured and semistructured data. In: Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on, pp. 543–546. IEEE (2008)

[148] Porres, I., Valiente, M.: Process Definition and Project Tracking in Model Driven Engineering. In: J. Münch, M. Vierimaa (eds.) Product-Focused Software Process Improvement, *Lecture Notes in Computer Science*, vol. 4034, pp. 127–141. Springer Berlin / Heidelberg (2006)

[149] Quispe, A.: An MDE Approach to Software Process Tailoring. In: Proceeding of the 2nd workshop on Software engineering for sensor network applications, pp. 43–52. ACM (2011)

[150] Riccobene, E., Scandurra, P., Rosti, A., Bocchio, S.: Designing a Unified Process for Embedded Systems. In: Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'07), pp. 77–90. IEEE (2007)

[151] Robles, G., González-Barahona, J.M.: A comprehensive study of software forks: Dates, reasons and outcomes. In: Open Source Systems: Long-Term Sustainability, pp. 1–14. Springer (2012)

[152] Rodríguez, D., García, E., Sánchez, S.: Defining Software Process Model Constraints with rules using OWL and SWRL. Int. J. Soft. Eng. Knowl. (2010)

[153] Rodriguez-Ellas, O., Martinez-Garcia, A.: Modeling and analysis of knowledge flows in software processes through the extension of the software process engineering metamodel. Knowledge Engineering (2009)

[154] Rosado, D.G., Fernández-Medina, E., López, J., Piattini, M.: Analysis of Secure Mobile Grid Systems: A systematic approach. Information and Software Technology **52**(5), 517–536 (2010)

[155] Rougemaille, S., Migeon, F., Millan, T., Gleizes, M.P.: Methodology Fragments Definition in SPEM for Designing Adaptive Methodology: A First Step. In: M. Luck, J. Gomez-Sanz (eds.) Agent-Oriented Software Engineering IX, *Lecture Notes in Computer Science*, vol. 5386, pp. 74–85. Springer Berlin / Heidelberg (2009)

[156] Ruiz, M., Pelechano, V.: Model Driven Design of Web Service Operations using Web Engineering Practices. In: M. Calisti, M. Walliser, S. Brantschen, M. Herbstritt, C. Pautasso, C. Bussler (eds.) Emerging Web Services Technology, Whitestein Series in Software Agent Technologies and Autonomic Computing, pp. 83–100. Springer (2007)

[157] Ruiz, M., Valderas, P., Pelechano, V.: Providing Methodological Support to Incorporate Presentation Properties in the Development of Web Services. In: G. Psaila, R. Wagner (eds.) E-Commerce and Web Technologies, *Lecture Notes in Computer Science*, vol. 4655, pp. 139–148. Springer Berlin / Heidelberg (2007)

[158] Ruiz-Rube, I., Cornejo, C.M., Dodero, J.M.: Accessing learning resources described in semantically enriched weblogs. International Journal of Metadata, Semantics and Ontologies **6**(3), 175–184 (2011)

[159] Ruiz-Rube, I., Cornejo-Crespo, C., Dodero, J.M., Ruiz, M.: Evaluación de un ecosistema software en organizaciones de desarrollo web bajo cmmi. In: Actas de las Jornadas de Ingeniería del Software y Bases de Datos, pp. 237–248 (2010)

[160] Ruiz-Rube, I., Dodero, J.M., Palomo-Duarte, M., Ruiz, M., Gawn, D.: Uses and applications of software & systems process engineering metamodel process models. a systematic mapping study. Journal of Software: Evolution and Process (2013)

[161] Ruiz-Rube, I., Dodero, J.M., Stoitsis, J.: Non-functional aspects of information integration and research for the web science. Procedia Computer Science **4**, 1631–1639 (2011)

[162] Saadawi, H., Wainer, G.A.: On the verification of hybrid devs models. In: Proceedings of 2012 Spring Simulation Conference (SpringSim12), DEVS/TMS Symposium, p. TBD. Society for Modeling and Simulation International (SCS), ACM (2012)

[163] Sabil, S., Jawawi, D.N.A.: Integration of PECOS into MARMOT for Embedded Real Time Software Component-Based Development. In: 2009 Fourth International Conference on Software Engineering Advances, pp. 265–270. IEEE (2009)

[164] Sampaio, A., Vasconcelos, A., Sampaio, P.: XWebProcess: Agile Software Development for Web Applications. In: N. Koch, P. Fraternali, M. Wirsing (eds.) Web Engineering, *Lecture Notes in Computer Science*, vol. 3140, p. 777. Springer Berlin / Heidelberg (2004)

[165] Schwaber, K., Beedle, M.: Agile software development with Scrum, vol. 1. Prentice Hall Upper Saddle River (2002)

[166] Scott, K.: The unified process explained. Addison-Wesley Longman Publishing Co., Inc. (2002)

[167] Scott Hawker, J., Hong, M., Smith, R.: A Web-based process and process models to find and deliver information to improve the quality of flight software. In: 22nd Digital Avionics Systems Conference Proceedings (Cat No 03CH37449) DASC-03, pp. 3.B.3–31. IEEE (2003)

[168] Seebach, H., Nafz, F., Steghofer, J.P., Reif, W.: A Software Engineering Guideline for Self-Organizing Resource-Flow Systems. In: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, pp. 194–203. IEEE (2010)

[169] SEI: CMMI® for Development, Version 1.2. Tech. Rep. August, Software Engineering Institute (2006)

[170] Seidita, V., Cossentino, M., Gaglio, S.: Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. In: M. Luck, J. Gomez-Sanz (eds.) Agent-Oriented Software Engineering IX, *Lecture Notes in Computer Science*, vol. 5386, pp. 46–59. Springer Berlin / Heidelberg (2009)

[171] Sethanandha, B., Massey, B., Jones, W.: Managing open source contributions for software project sustainability. In: Technology Management for Global Economic Growth (PICMET), 2010 Proceedings of PICMET '10:, pp. 1–9. IEEE (2010)

[172] Shirabad, JS. and Menzies, T.: PROMISE Software Engineering Repository (2005)

[173] Silingas, D., Pavalkis, S., Morkevicius, A.: MD wizard - a model-driven framework for wizard-based modeling guidance in UML tools. In: 2009 International Multiconference on Computer Science and Information Technology, pp. 609–615. IEEE (2009)

[174] Silva, M., Oliveira, T.: Towards Detailed Software Artifact Specification with SPEMArti. In: Proceeding of the 2nd workshop on Software engineering for sensor network applications, pp. 213–217. ACM (2011)

[175] Singh, R.: International Standard ISO/IEC 12207 Software Life Cycle Processes. Software Process: Improvement and Practice **2**(1), 35–50 (1996)

[176] Sousa, K., Mendonça, H., Vanderdonckt, J.: Towards Method Engineering of Model-Driven User Interface Development. In: M. Winckler, H. Johnson, P. Palanque (eds.) Task Models and Diagrams for User Interface Design, *Lecture Notes in Computer Science*, vol. 4849, pp. 112–125. Springer Berlin / Heidelberg (2007)

[177] Stahl, T.T., Voelter, M.: Model-driven software development. John Wiley & Sons Chichester (2006)

[178] Sudeikat, J., Renz, W.: On the Modeling, Refinement and Integration of Decentralized Agent Coordination. In: D. Weyns, S. Malek, R. de Lemos, J. Andersson (eds.) Self-Organizing Architectures, *Lecture Notes in Computer Science*, vol. 6090, pp. 251–274. Springer Berlin / Heidelberg (2010)

[179] Theunissen, W.H.M., Boake, A., Kourie, D.: In search of the sweet spot: agile open collaborative corporate software development. In: Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, SAICSIT '05, pp. 268–277. ACM (2005)

[180] Tran, H.N., Coulette, B., Tran, D.T., Vu, M.H.: Automatic Reuse of Process Patterns in Process Modeling. In: Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 1431–1438. ACM (2011)

[181] Tranoris, C., Thramboulidis, K.: A tool supported engineering process for developing control applications. Computers in Industry **57**(5), 462–472 (2006)

[182] Traverso-Ribón, I., Ruíz-Rube, I., Dodero, J.M., Palomo-Duarte, M.: Open data framework for sustainable assessment in software forges. In: Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics, p. 20. ACM (2013)

[183] Trujillo, J., Soler, E., Fernández-Medina, E., Piattini, M.: An engineering process for developing Secure Data Warehouses. Information and Software Technology **51**(6), 1033–1051 (2009)

[184] Tu, M., That, T., Coulette, B., Lbath, R., Tran, H.N., Nassar, M.: Towards a tool-supported approach for collaborative process modeling and enactment. In: 2011 18th Asia-Pacific Software Engineering Conference. IEEE (2011)

[185] Turner, M.S.V.: Microsoft Solutions Framework Essentials. O'Reilly Media, Inc. (2009)

[186] Valderas, P., Fons, J., Vera, C.D.: Allowing End-users to Participate within Model-Driven Development Approaches. In: Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on, pp. 187–190. IEEE (2011)

[187] Valderas, P., Pelechano, V., Pastor, O.: Introducing Graphic Designers in a Web Development Process. In: J. Krogstie, A. Opdahl, G. Sindre (eds.) Advanced Information Systems Engineering, *Lecture Notes in Computer Science*, vol. 4495, pp. 395–408. Springer Berlin / Heidelberg (2007)

[188] Vieira, V., Tedesco, P., Salgado, A.C.: Designing context-sensitive systems: An integrated approach. Expert Systems with Applications **38**(2), 1119–1138 (2011)

[189] Vlissides, J., Helm, R., Johnson, R., Gamma, E.: Design patterns: Elements of reusable object-oriented software. Reading: Addison-Wesley (1995)

[190] Washizaki, H.: Deriving Project-Specific Processes from Process Line Architecture with Commonality and Variability. In: 2006 IEEE International Conference on Industrial Informatics, pp. 1301–1306. IEEE (2006)

[191] Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag Berlin Heidelberg (2007)

[192] Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering **11**(1), 102–107 (2005)

[193] Wookjin, L., Sanghyun, P., Keeyoull, L., Chunwoo, L., Byungjeong, L., Woosung, J., Taeksu, K., Heechern, K., Chisu, W.: Agile Development of Web Application by Supporting Process Execution and Extended UML Model. In: 12th Asia-Pacific Software Engineering Conference (APSEC'05), pp. 193–200. IEEE (2005)

[194] Workgroup, O.C.S.: Oslc core specification version 3.0 draft. Tech. rep., OSLC (2013)

[195] Yatchou, R., Nkambou, R., Tangha, C.: Intelligent Learning Environment for Software Engineering Processes. In: J.C. Lester, R.M. Vicari, F. Paraguaçu (eds.) Intelligent Tutoring Systems, *Lecture Notes in Computer Science*, vol. 3220, pp. 898–900. Springer Berlin / Heidelberg (2004)

[196] Yoonjung, C., Su-Jung, H., Jin-Sam, K.: Eclipse-based management system for process innovation & methodology enhancement. In: 2006 8th International Conference Advanced Communication Technology, pp. 5 pp.–159. IEEE (2006)

[197] Zhu, L., Tran, T., Staples, M., Jeffery, R.: Technical Software Development Process in the XML Domain. In: Q. Wang, V. Garousi, R. Madachy, D. Pfahl (eds.) Trustworthy Software Development Processes, *Lecture Notes in Computer Science*, vol. 5543, pp. 246–255. Springer Berlin / Heidelberg (2009)

La Ingeniería de Procesos Software promueve la producción sistemática de software mediante el seguimiento de una serie de procesos técnicos y de gestión bien definidos. La gestión integral de los procesos implica el modelado, análisis, despliegue y evaluación de los procesos. Sin embargo, su automatización requiere de mayores esfuerzos de investigación y desarrollo.

En esta tesis doctoral se presenta *Software Process Deployment & Evaluation Framework* (SPDEF), un marco de trabajo para el despliegue y evaluación de procesos sobre herramientas de soporte a la gestión o producción del software. Este framework incluye un método para la adaptación automática de las herramientas, mediante la aplicación de técnicas de la Ingeniería del Software Dirigida por Modelos, y para la construcción de soluciones de integración de información para la evaluación de los procesos, mediante el enfoque de Datos Abiertos Enlazados. El framework incluye, además de un método sistemático para el despliegue y evaluación, un conjunto de modelos y relaciones, así como una serie de herramientas de apoyo. De cara a la evaluación del framework se presentan dos casos de estudio y un escenario descriptivo de utilización.

Memoria de tesis doctoral presentada en Diciembre de 2013 como parte del Doctorado en Ingeniería y Arquitectura de la Universidad de Cádiz.

Escuela Superior de Ingeniería
Departamento de Ingeniería Informática
Grupo: Mejora del Proceso Software y Métodos Formales